National Defence / Défense nationale

UNCLASSIFIED

UNLIMITED DISTRIBUTION

DRES

AD-A239 385

SUFFIELD MEMORANDUM

NO. 1353

A RESEARCH CODE TO STUDY SOLUTIONS OF THE BOUNDARY LAYER EQUATIONS IN BODY CONFORMAL COORDINATES

by

Denis Bergeron

May 1991

DEFENCE RESEARCH ESTABLISHMENT SUFFIELD, RALSTON, ALBERTA

Canada

Defence Research Establishment Suffield
Ralston, Alberta

Suffield Memorandum No. 1353

A Research Code to Study Solutions of the Boundary Layer Equations in Body Conformal Coordinates

by

Denis Bergeron

ABSTRACT

This report describes the development of a computer code to solve the two-dimensional boundary layer equations in direct, inverse or mixed direct/inverse mode for airfoil flows. The solution algorithm uses body conformal coordinates and a relaxation scheme with flow-dependent operators. The code incorporates two methods to sweep the flow field.

Topics include a description of the code structure, its input requirements, boundary condition and flow field initialization and various software tools required by the main algorithm. Finally, verification results are presented.

This work, done in cooperation with the University of Toronto, seeks the development of a boundary layer code compatible with the NASA Ames ARC2D Navier-Stokes code. The two codes will be used in a study of the Fortified Navier-Stokes concept.

RÉSUMÉ

Ce rapport décrit le développement d'un logiciel pour résoudre les équations des couches limites appliquées à l'écoulement autour d'un profil d'aile. Les équations sont exprimées en coordonnées conformes au corps étudié; une méthode de relaxation avec des opérateurs s'ajustant à l'écoulement local ainsi que des méthodes de mode direct, inverse et mixte direct/inverse sont employées. Enfin, deux méthodes de balayage du champ sont utilisées.

Les sujets suivants sont abordés: la structure du logiciel, les entrées nécessaires à son opération, les conditions limites et l'initiation des variables. De plus, différents outils numériques requis par le logiciel principal sont décrits et des résultats démontrant la validité du lociciel sont présentés.

Ce travail, mené en collaboration avec l'Université de Toronto, vise à élaborer un programme de couche limite compatible avec le programme Navier-Stokes ARC2D mis au point par le centre de recherche Ames de la NASA. Les deux programmes serviront à étudier le concept fortifié de Navier-Stokes.

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

a	Speed of sound
A_k	Lower diagonal coefficients of tridiagonal system of equations
$[A]$	Tridiagonal or modified tridiagonal matrix
A^+	Constant used in turbulence model
B_k	Diagonal coefficients of tridiagonal system of equations
$\vec{B}$	Right hand side vector of tridiagonal system of equations
c_p	Coefficient of specific heat at constant pressure
c_v	Coefficient of specific heat at constant volume
C_k	Upper diagonal coefficients of tridiagonal system of equations
D_k	Right hand side coefficients of tridiagonal system of equations
f	Falkner-Skan stream function
f_l , f_u	Scale factor for lower and upper points of grid generator
F_k	First column coefficients of modified tridiagonal system of equations
F_{Kleb}	Klebanoff intermittency factor in turbulence model
$F_{\max}$	Maximum of function $F(n)$ in turbulence model
F_{wake}	Scaling factor used in turbulence model
H	Total enthalpy, $H = c_p + u^2/2$
l	Reference length
m	Self-similarity parameter for Falkner-Skan flows
$\bar{m}$	Non-dimensional pressure gradient for Klineberg flow
M	Mach number
p	Pressure
Pr	Prandtl number
R	Gas constant
Re	Ratio of freestream Reynolds and Mach numbers, $\mathrm{Re} = \rho_\infty l a_\infty / \mu_\infty$
s,n	Body conformal coordinates in the streamwise and normal directions
S_1	Constant used in Sutherland's approximation for viscosity
t	Time-like variable used to relax the equations

T	Absolute temperature
u, υ	Body-conformal velocity components
u^+, n^+	Boundary layer coordinates used for Andersen flow
u_{dif}	Difference of minimum and maximum velocity in turbulence model
U	Contravariant velocity in the steamwise direction
x, y	Cartesian coordinates
$\vec{X}$	vector of unknowns of tridiagonal system of equations

Greek Symbols

α	Falkner-Skan coefficient or angle-of-attack of airfoil
β	Falkner-Skan coefficient, related to self-similarity parameter m
γ	Ratio of specific heats, c_p / c_v
δ^*	Boundary layer displacement thickness
Δ	Indicates small difference of a variable, e.g. Δt
ε	Expansion factor for grid generation
ζ	Non-dimensional height in Falkner-Skan equation
ξ, η	Coordinates of computational plane
θ	Rotation angle for Cartesian to body conformal transform
μ	Viscosity
ν	Kinematic viscosity
ρ	Density
τ	Shear stress, $\tau = \mu(\partial u / \partial y)$
ω	Vorticity

Superscripts

$(\tilde{\ })$	Non-dimensionalized value
$(\ )^i$	Dummy index for inverse function update
$(\ )^n$	Dummy time index

Subscripts

$(\ )_b$	Body conformal component
$(\ )_c$	Cartesian component
$(\ )_e$	Edge value in η direction, or end point for grid generation
$(\ )_j, (\ )_k$	Dummy spacial indices
$(\ )_m$	Molecular value
$(\ )_t$	Turbulent value
$(\ )_{SL}$	Reference conditions in Sutherland's approximation
$(\ )_w$	Value at a solid boundary (wall)
$(\ )_\infty$	Freestream values
$(\ )_2$	From $\eta = 2$ point
$(\ )_{KBL}$	From $\eta = KBL$ point

Operators

E_η^{+1}	Shift operator in η direction
Δ_η	First order backward finite difference operator

INTRODUCTION

Background

In the mid 1980's, researchers at NASA Ames proposed the Fortified Navier-Stokes (FNS) concept, a procedure by which approximate solutions, or any other known information, is used to improve the accuracy and efficiency of a Navier-Stokes code. Between 1985 and 1988, Steger and Van Dalsem[1][2][3] explored the FNS concept by combining a Navier-Stokes code with a boundary layer code. The boundary layer equations are solved in a body conformal frame of reference while the Navier-Stokes solver uses Cartesian velocity components with generalized curvilinear coordinates. Since the two procedures use different grids and velocity components, going from one solution to the other involves interpolation and rotation of the velocity data. Despite these differences, Steger and van Dalsem still improved the convergence of their Navier-Stokes code by an order of magnitude. Their work laid a solid foundation for further investigations of the FNS concept.

The purpose of the current project is to investigate the range of validity of the FNS approach as a function of factors like Mach number, Reynolds number, and angle of attack. Three phases are planned:

i. the development of a boundary layer solver compatible with the NASA Ames code ARC2D,

ii. integration of the boundary layer and Navier-Stokes codes, and

iii. parametric studies using the FNS code.

This report presents partial results for phase i) above. It describes the development of GBL, a boundary layer research code to study the Steger and Van Dalsem algorithms. GBL has the flexibility to support changes to the formulation of the equations, the boundary conditions, or to the various iterative schemes used to solve the equations.

At this point, it is useful to review the boundary layer work of Steger and Van Dalsem.

Previous Generalized Boundary Layer Work

Several boundary layer methods (e.g. Cebeci[4]) employ a coordinate transformation, or "stretching" of the governing equations prior to formulation of the finite difference equations. The purpose is to scale the growth of the viscous layer so that the new equations, expressed in terms of similarity variables, may be solved on an approximately rectangular grid. Van Dalsem and Steger[5] took a different approach. They developed a scheme to solve the two-dimensional, steady state boundary layer equations in body-

conformal coordinates (s,n), which amounts to solving the equations in physical - instead of similarity - variables. An adaptive grid must then be used to concentrate a sufficient number of computational points within the boundary layer. Solution of the equations on a grid with uneven spacing results in complicated finite difference equations, but this is avoided by transforming the equations to a computational domain with uniform grid spacing so that standard, unweighted differences can be used.

The scheme of Van Dalsem and Steger is noteworthy for two reasons. First, it assumes the equations are only weakly coupled when the pressure is given. Each equation can then be solved independently in a sequential scheme. They also chose their finite difference operators to obtain bidiagonal or tridiagonal systems of equations which are solved efficiently with the Thomas algorithm.

Second, their scheme avoids use of complex space marching procedures within separated flow regions. Instead, the scheme marches in a single direction while flow-dependent finite difference operators adapt locally to respect the parabolic nature of the boundary layer equations.

The earlier work of Steger and Van Dalsem[5] used a predictor-corrector finite difference approximation to the two-dimensional, steady-state equations. A mix of upwind and central operators is used to differentiate the convective terms within the momentum and energy equations. The choice of which operator to use depends on the local value of u/u_e. This algorithm was later expanded[6] to the three-dimensional unsteady form. The formulation was also simplified to a single-step process and only the upwind operator, dependent on the sign of the contravariant velocity U or W, was retained for the convective terms. The reader should be aware that Steger and Van Dalsem do not use the time variable in a true time fashion, since the method still assumes that the pressure distribution is fixed and given. Instead, the "time-like" variable is used to relax the equations.

The above review would not be complete without mention that, in 1988, Steger and Van Dalsem[7] developed a very different approach to solving the boundary layer equations. Having recognized the lack of a boundary layer procedure directly compatible with their Navier-Stokes algorithm, they recast the boundary layer equations in a form using the Cartesian velocity components instead of the usual body conformal components. The new equations can be solved on a Cartesian grid, thus, increasing commonality with Navier-Stokes codes.

GBL Program Structure

GBL uses the 1986 body conformal version of the Steger and Van Dalsem boundary layer equations with time-like relaxation. It was selected over the 1988 version because it is simpler to implement and provides a framework from which to study the behavior of

the algorithm. Scaling of the equations, however, was modified[8] to increase their compatibility with Pulliam's ARC2D[9] Navier-Stokes code. The latter will be used as the Navier-Stokes components of the FNS code.

The infrastructure of GBL is designed in four general parts: input, initialization, computations and output. In addition, specialized graphics tools were developed to control program execution and to examine data during the development cycle. However, these tools are machine specific and their description is outside the scope of this work. For these reasons, the graphics tools are not described in this report.

Program inputs are described in the PROGRAM INPUTS section. They include parameters specifying the freestream conditions as well as switches to control program execution. In addition, the user may elect to run one of the built-in test cases to verify the implementation of a new algorithm, or to run an airfoil flow field calculation. The user may also direct GBL to generate its own grids for the test cases or select one of two grid generation schemes for the airfoil case (an "adaptive" grid or a subset of the Navier-Stokes grid). Various tools are required by the input, initialization and computation routines as well as for output processing. These are described in the COMPUTATIONAL TOOLS section.

Four initialization procedures may be used by GBL: two use ARC2D data to initialize the flow about an airfoil while the others are test cases for flow over a flat plate. A description of these procedures is provided in the INITIALIZATION OF VARIABLES section. In the SOLUTION SCHEME section, the algorithm and overall program flow along with the application of boundary conditions, sweeping schemes and determination of the inverse mode region are described.

The CODE TESTING section present code validation results while conclusions are presented in the last section.

[BLANK PAGE]

PROGRAM INPUTS

The user selects the run parameters for GBL by editing the control file *inp/run.sw* which contains sufficient information for GBL to proceed automatically, whether it is a test case or airfoil computations. The parameters are divided into two general categories: essential parameters, which are always required by the program, and test parameters which are used to verify new algorithms, as listed below.

Essential parameters:

[1] Freestream parameters, used to compute all freestream variables required to scale the equations.

[2] Turbulence model switch and transition points for lower and upper surfaces of the airfoil.

[3] Grid parameters which include a switch to chose between a flat plate grid generated internally or an airfoil grid read from ARC2D. The flat plate grid generator requires user supplied specifications while for airfoil cases, the boundary layer grid is generated automatically from the Navier-Stokes grid. Grid generation is discussed in further details later in this section.

[4] Time step to use for the time-like relaxation scheme and total number of cycles for the run.

[5] Convergence criteria to use for computation of the residuals.

Test parameters:

[1] Mode (direct/inverse) to be tested.

[2] Direction for sweeps of the computational field during the solution process.

[3] Switches to control execution of individual equations during the solution process.

[4] Self-similarity parameter for test cases involving Falkner-Skan flow.

[5] Control over the ξ range of the computation; particularly useful when investigating behavior of the algorithm over a specified flow region, e.g. about the airfoil nose or in its wake.

[6] Control over the extent of the region where the inverse formulation is used as well as specification of parameters used to update the inverse forcing function (to be explained later).

[7] Control over individual boundary conditions for the momentum and energy equations.

[8] Selection of an additional relaxation factor (extra to the time-like terms) which may be applied to the solution of individual equations.

A sample control file is given in appendix A along with a brief description of each parameter.

GBL supports flow field initialization for several test and airfoil cases. The test cases involve Falkner-Skan or Klineberg flow. These flows are stricly laminar, however, the Falkner-Skan case may also be used with turbulence to simulate turbulent flat plate flow. For either case, GBL generates a flat plate boundary layer grid according to user specifications. Airfoil cases require the user to provide a Navier-Stokes grid and the corresponding state vector from ARC2D. GBL then generates a compatible boundary layer grid and initializes flow field variables from the ARC2D data.

The remainder of this section provides details about computation of the freestream parameters, generation of the flat plate and airfoil boundary layer grids and their transformation to body conformal coordinates.

Freestream Parameters

The freestream parameters are used to scale flow field variables. The user provides four parameters: (i) the Reynolds number (Re_∞), (ii) the Mach number (M_∞), (iii) the freestream temperature (T_∞) and (iv) a reference length (l). All other freestream parameters are computed from these four values with the assumption that the fluid behaves like a perfect gas and that the flow is isentropic. The computations proceed in the following order.

The freestream speed of sound is calculated from the freestream temperature, the ratio of specific heat at constant pressure to specific heat at constant volume, γ (1.4 for air), and the gas constant R (287.3 $m^2/K\text{-}sec^2$ for air).

$$a_\infty = (\gamma R T_\infty)^{1/2} \tag{2-1}$$

Having determined a_∞, the freestream velocity u_∞ is obtained from the Mach number definition

$$u_\infty = M_\infty a_\infty \tag{2-2}$$

The freestream temperature is also used with Sutherland's approximation[10] (p. 312) and reference values of the temperature and viscosity at sea level, as defined in the standard

atmosphere ($T_{SL} = 288.16K$, $\mu_{SL} = 1.78935\times10^{-5}\ N-sec/m^2$), to compute freestream viscosity

$$\mu_\infty = \frac{1.458\times10^{-6}\, T_\infty^{3/2}}{T_\infty + S_1} \tag{2-3}$$

where $S_1 = 110.4K$. Freestream density is obtained from the definition of the freestream Reynolds number and the reference length

$$\rho_\infty = \frac{Re_\infty \mu_\infty}{u_\infty l} \tag{2-4}$$

while the kinematic viscosity is simply the ratio of viscosity to density

$$\nu_\infty = \frac{\mu_\infty}{\rho_\infty} \tag{2-5}$$

and freestream pressure is computed from the perfect gas relation

$$p_\infty = \rho_\infty R T_\infty \tag{2-6}$$

Finally, freestream total enthalpy is computed by substituting freestream values in its definition

$$H_\infty = c_p T + \frac{u_\infty^2}{2} \tag{2-7}$$

Note that the normal velocity v_∞ is omitted from the above definition of total enthalpy because of the body conformal assumption used to derive the equations.

Flat Plate Grid Generation

GBL incorporates two sample cases to test the coding accuracy and/or the general behavior of present and future algorithms. The first test case simulates self-similar Falkner-Skan flow while the second case simulates a linearly decelerated flow with mild

separation, initially computed by Klineberg and Steger. Both test cases, which are explained in detail later in this report, require a flat plate grid generated internally by GBL according to user specifications.

The run parameter file contains the following flat plate grid information: start coordinate $\tilde{s}_s$, end coordinate $\tilde{s}_e$, the total number of streamwise stations NXS, and the number of points normal to the streamwise direction at each station KBL. GBL assumes that the stagnation point is located at the $\tilde{s} = 0$ position and that $\tilde{s}_s$ and $\tilde{s}_e$ are positive numbers.

At each station, the first point above the flat plate surface, $\tilde{n}_2$, is located at $n^+ = 1$, based on the Prandlt logarithmic law of the wall, and multiplied by the user selected factor f_l. Freestream conditions and the theoretical Blasius wall shear stress distribution are used in the computation. Different values are obtained for laminar and turbulent flow as follows:

$$\tilde{n}_2 = \frac{f_l}{\mathrm{Re}_\infty (\tilde{\tau}_w)^{1/2}} \tag{2-8a}$$

and the non dimensional wall shear stress $\tilde{\tau}_w$ takes different values for laminar and turbulent flow. For laminar flow, it is defined as

$$\tilde{\tau}_{w_l} = 0.332 M_\infty^2 \left[\frac{\nu_\infty}{u_\infty \tilde{s} l} \right]^{1/2} \tag{2-8b}$$

while for turbulent flow, it is given by

$$\tilde{\tau}_{w_t} = 0.0225 M_\infty^2 \left[\frac{\nu_\infty}{8 \delta^* u_\infty} \right]^{1/4} \tag{2-8c}$$

where δ^* is the turbulent flat plate boundary layer displacement thickness (see reference 10, p.599)

$$\delta^* = 0.125 \times \left[0.37 \tilde{s} l \left[\frac{u_\infty \tilde{s} l}{\nu_\infty} \right]^{-1/5} \right] \tag{2-9}$$

For each station, the upper boundary in the $\tilde{n}$ direction, $\tilde{n}_{KBL}$, is located at the corresponding Blasius displacement thickness multiplied by an amplification factor f_u of

the order of 1.5 to 5. For laminar flow, this yields

$$\tilde{n}_{KBL} = f_u\, 5.0 \left[\frac{\nu_\infty \tilde{s} l}{u_\infty} \right]^{1/2} \tag{2-10a}$$

while for turbulent flow, we have

$$\tilde{n}_{KBL} = f_u\, \delta^* \tag{2-10b}$$

The location of the KBL-3 points between $\tilde{n}_2$ and $\tilde{n}_{KBL}$ is determined from the exponential stretching function

$$\tilde{n}_k = \tilde{n}_{k-1} + \tilde{n}_2 (1+\varepsilon)^{(k-2)} \qquad k = 3, KBL-1 \tag{2-11}$$

where ε is computed using a Newton-Raphson method to yield $\tilde{n}_{KBL}$ when $k = \text{KBL}$. Figure 1 shows a typical flat plate grid generated by GBL.

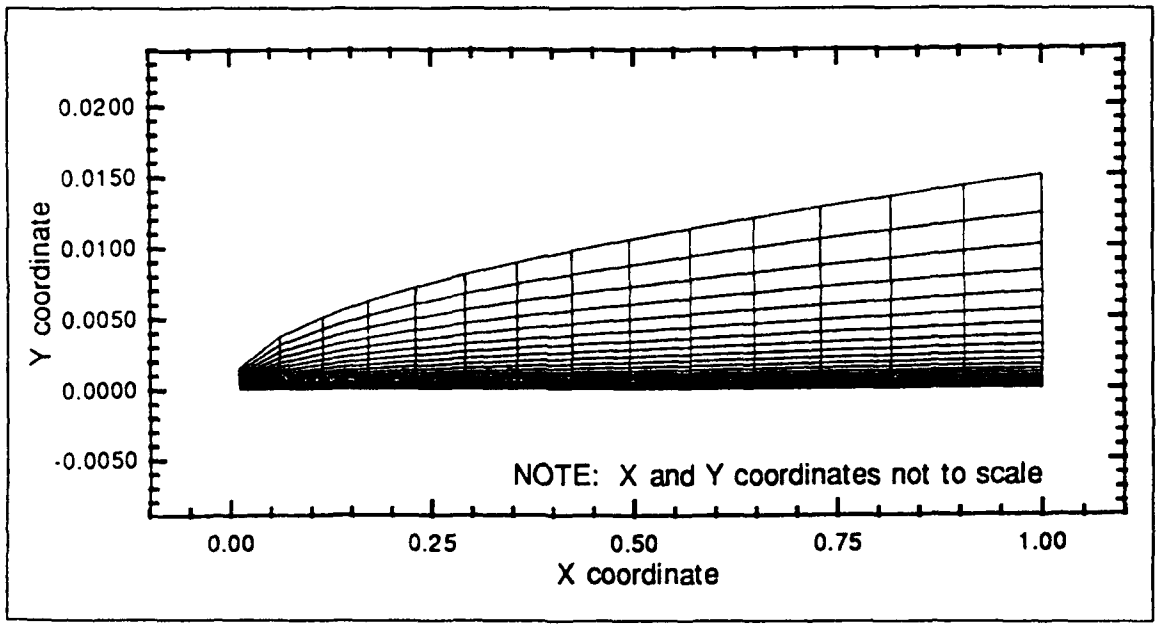

Figure 1. Flat plate grid generated by GBL

Navier-Stokes and Boundary Layer Grids

When computing airfoil flow, GBL requires two files from the ARC2D Navier-Stokes code: the grid file which contains the Cartesian coordinates $(\tilde{x},\tilde{y})$ of the numerical grid, and the state vector file which contains the density, $\tilde{x}$-momentum, $\tilde{y}$-momentum, and energy information for each grid point.

Only grids of C-type topology, as shown in figure 2, are used in the present study. These are generated with a hyperbolic grid generation program. Each η line starts from the lower far-field wake, wraps around the airfoil, and ends at the upper far wake to form a "C" shape. The ξ lines, perpendicular to the η lines, extend from the wake centerline, or airfoil surface, to the far field outer boundary. The line defining the airfoil surface and the wake centerline is the $\eta = 1$ line. The $\xi = 1$ line is the lower far wake boundary. An effort is made to keep the orthogonality of each (ξ,η) line intersection. This is particularly important near a solid boundary or at the wake centerline where we want to solve the boundary layer equations. The reader should also notice that, for a Navier-Stokes grid, the η grid lines are concentrated where viscous effects dominate the flow, i.e. near the airfoil surface and wake centerline.

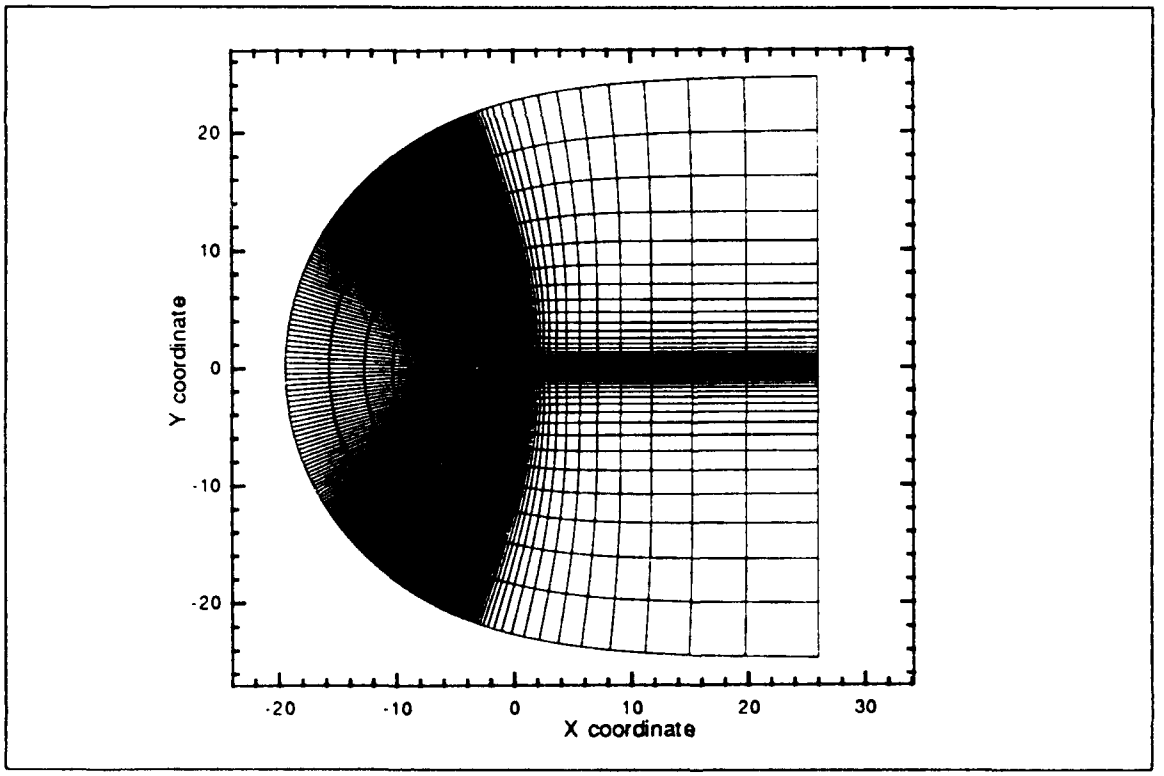

Figure 2. Navier-Stokes grid in Cartesian coordinates

Compared to the chord length, the thickness of an airfoil boundary layer is thin, growing from a fraction of a percent near the leading edge stagnation point, to some five or ten

percent at the trailing edge. It is then reasonable to assume that points near the airfoil surface, for a given ξ station, are lined up on the normal to the surface at that same station. A close examination of the Navier-Stokes grid shows this assumption to be valid over most of the airfoil except for the trailing edge (see figure 3). At that point, three line segments intersect: two from the airfoil lower and upper surfaces respectively, and one from the wake centerline. This causes a significant slope discontinuity which makes it impossible to define a normal to both the airfoil surface and the wake centerline; an average normal is used instead. Hence, one must be careful in using trailing edge Navier-Stokes coordinates for the boundary layer solution, since they are not truly normal to the surface. Furthermore, the ξ lines are normally clustered at the trailing edge to resolve local velocity gradients, thus several lines may not be normal to the surface in that region.

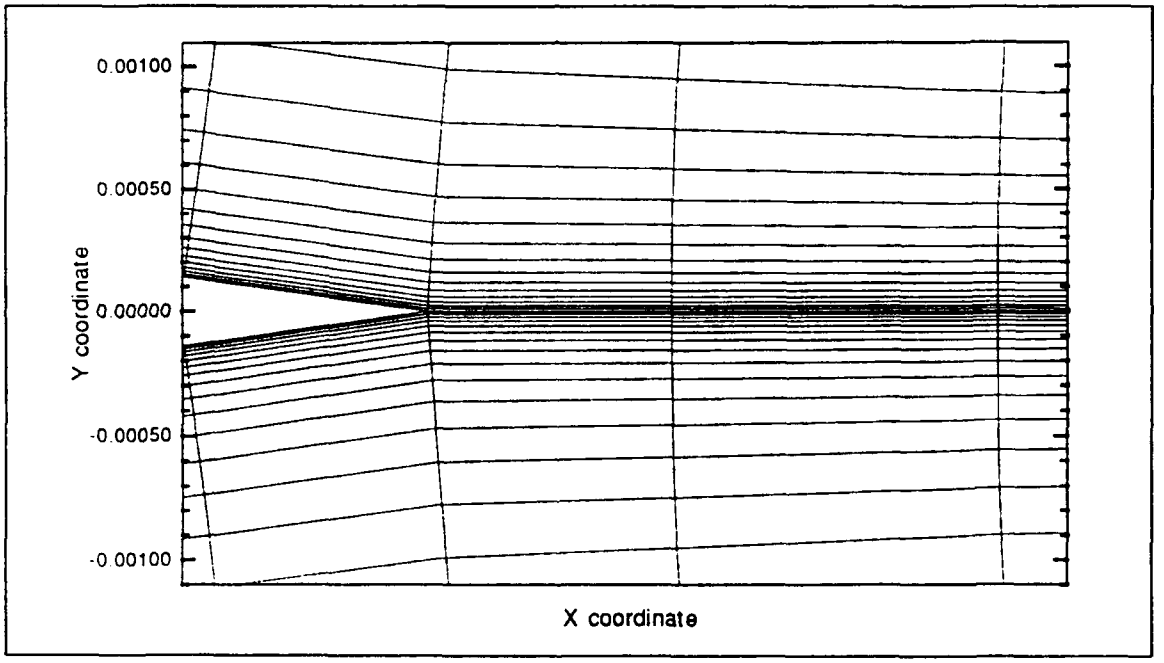

Figure 3. Loss of orthogonality at the airfoil trailing edge

GBL uses the ARC2D Cartesian coordinates $(\tilde{x},\tilde{y})$ to generate one of two possible boundary layer grids. The first grid type consists of a subset from the Navier-Stokes grid while the second grid type, discussed later in this section, is an "adaptive" grid.

Generating the boundary layer grid (see figure 4) from the Navier-Stokes grid presents the advantage that the Navier-Stokes and boundary layer procedures share the same grid. Interpolation of the variables is avoided and the overall computing expense of the FNS procedure is reduced. However, to be useful as a boundary layer grid, a subset of the Navier-Stokes grid grid must have a sufficient concentration of grid lines near the airfoil surface to resolve the strong normal gradients characterizing the boundary layer region. This is particularly important near the leading edge where the boundary layer is very

thin. On the other hand, typical Navier-Stokes grids are generated such that the normal distance between each η line and the airfoil surface is approximately constant. This means only a small number of points span the boundary layer near the leading edge, while more points than required span the boundary layer at the trailing edge. At least ten points should span the boundary layer near the leading edge while no more than 35 to 45 points are required for the trailing edge. Fortunately, the rapid stretching of the normal coordinates used to generate the Navier-Stokes grid respects the above conditions on the number of points across the boundary layer.

The boundary layer grid of figure 4 is used only partially by GBL. For each ξ station, GBL determines the approximate number of grid points that span the boundary layer and stores it in the grid index array NDT(). This is done for two reasons. First, why waste computer power on grid points which are clearly outside the boundary layer? There is however a glitch with this approach. The finite difference algorithm requires the streamwise velocity from adjacent stations to compute the convection derivatives, and at times, these values are from points outside the boundary layer. It is then necessary to update the value of points outside the NDT() boundary. For the momentum and energy equations, outside values are set equal to the value at NDT(). In most cases, this is acceptable because the NDT() boundary varies smoothly from station to station.

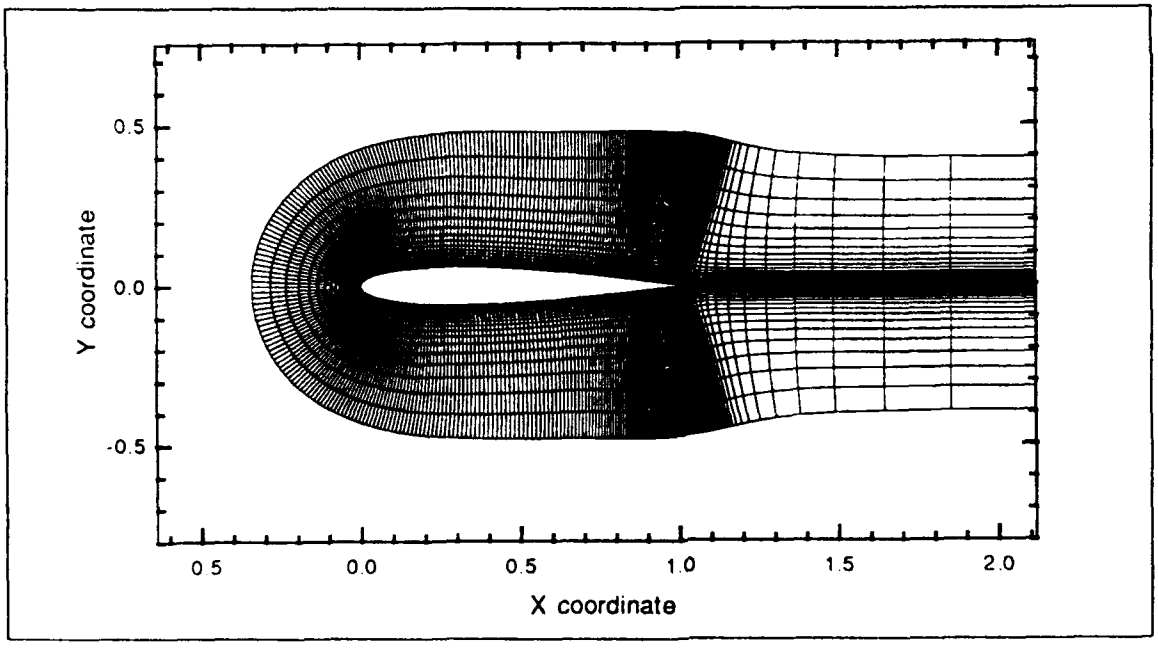

Figure 4. Boundary layer grid, in Cartesian coordinates, formed from a subset of the Navier-Stokes grid.

The second reason to restrain the computations to NDT() points only is related to the continuity equation. The boundary layer equations are not applicable to the flow outside the boundary layers and wake, where they predict constant streamwise velocity gradients

in the outer flow. Integration of these gradients with the continuity equation yields unbounded normal velocity $\tilde{v}$ and leads to instability of the numerical algorithm. It is then necessary to neglect the values of $\tilde{v}$ outside the NDT() boundary.

Adaptive Boundary Layer Grid

The previous grid has only a small number of grid points in the leading edge region. This may affect the accuracy in the upstream region and the resulting errors are propagated downstream due to the parabolic nature of the equations. An "adaptive" grid, where the word adaptive indicates that scaling of the grid is such that an approximately constant number of points span the physical boundary layer at any station, may increase the accuracy in the leading edge region and improve the overall accuracy of the code.

Several schemes were studied to generate an adaptive boundary layer grid. The one selected uses the Navier-Stokes grid as an underlay. This ensures the orthogonality characteristics of the Navier-Stokes grid are retained in the boundary layer grid, an important fact near the trailing edge. The overlay property also reduces the interpolation of the Navier-Stokes data onto the boundary layer points to a single dimension. An example grid is shown in figure 5.

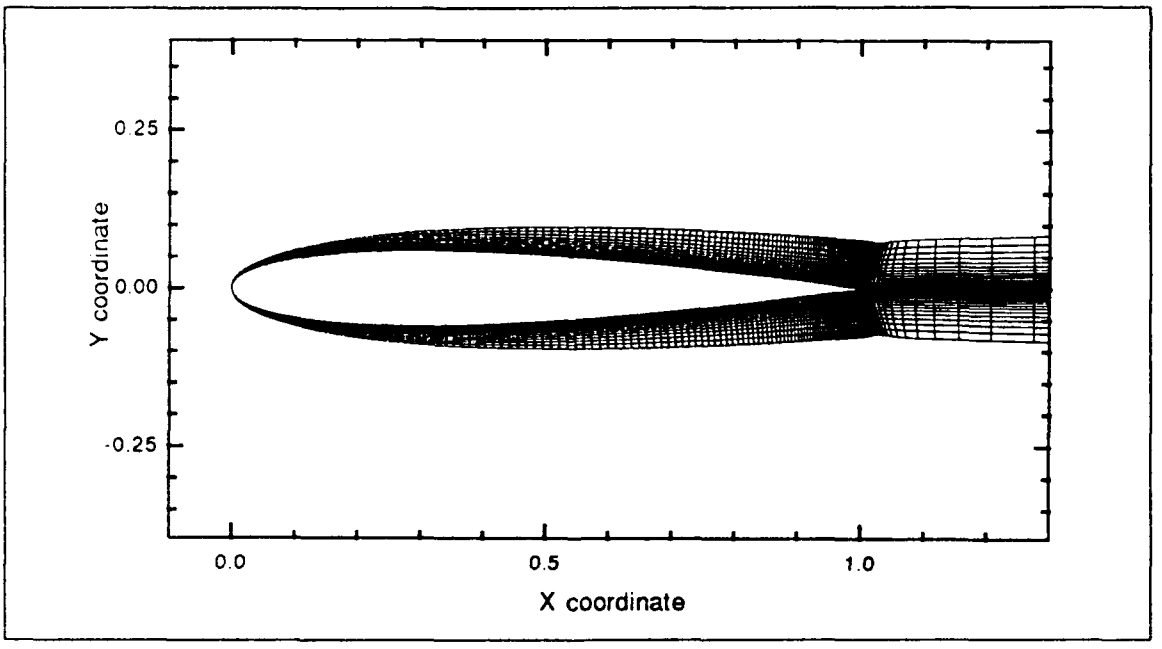

Figure 5. Adaptive boundary layer grid in Cartesian coordinates

The upper limit of the adaptive grid, in the direction normal to the surface, is generated from the turbulent flat plate boundary layer thickness distribution of equation (2-9) and

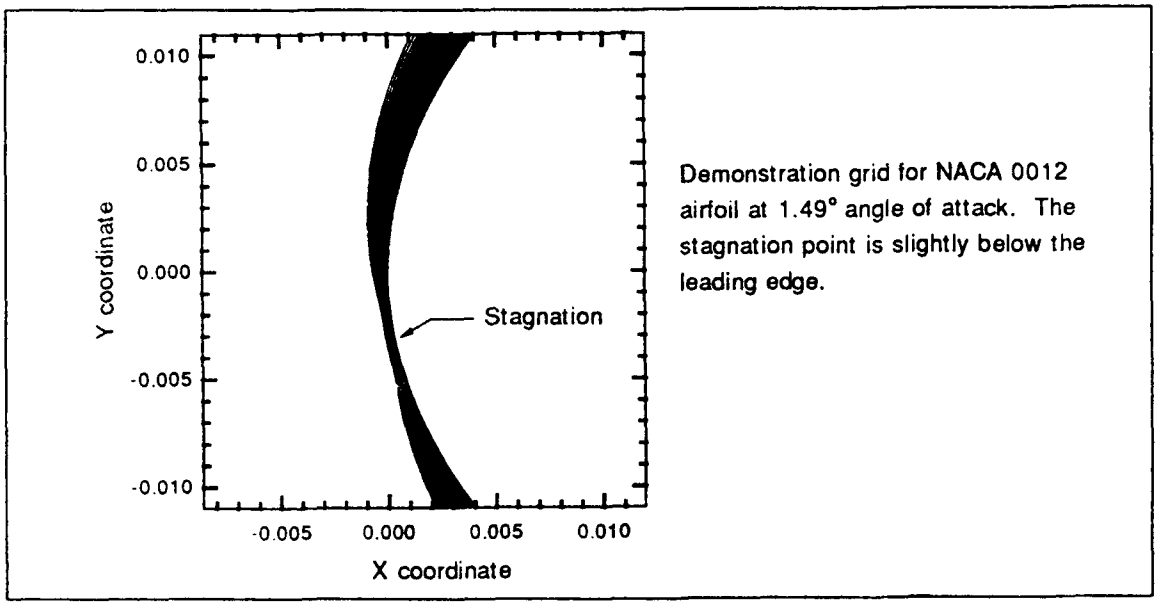

Figure 6. Adaptive grid details near the leading edge

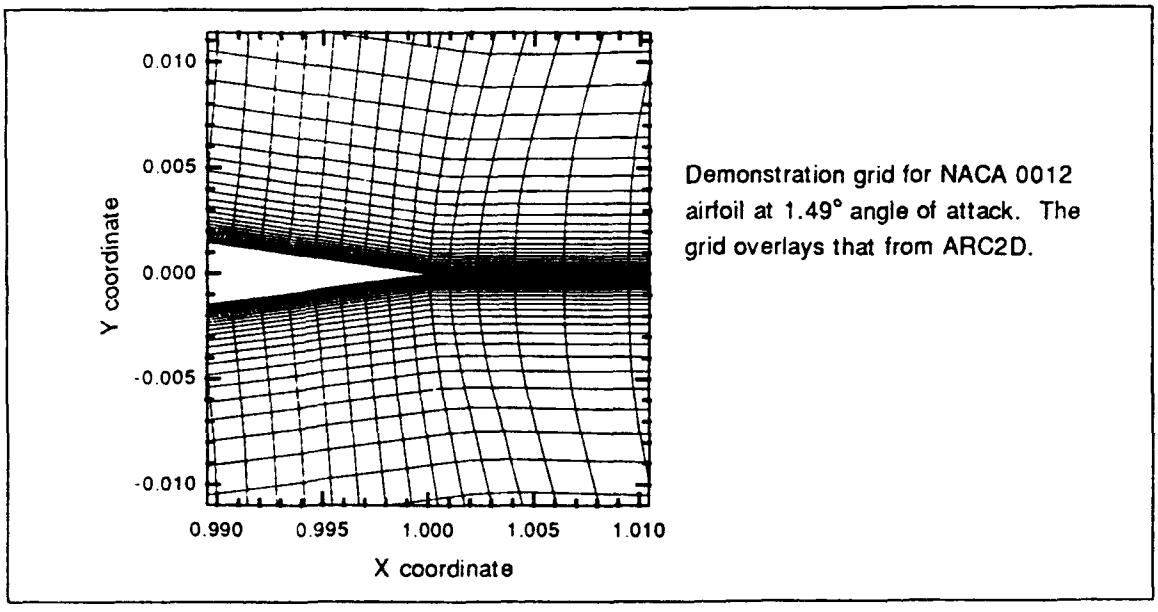

Figure 7. Adaptive grid details near the trailing edge

the appropriate freestream conditions. This is adequate to approximate the growth of the boundary layer over the airfoil. Recall that a flat plate experiences zero pressure gradient throughout while an airfoil experiences a strong negative pressure gradient over its forward portion followed by a positive pressure gradient, sually starting near :e point of maximum thickness. Therefore, the flat plate distribution overestimates the growth of the boundary layer near the airfoil leading edge while the opposite is true aft of the point of maximum thickness. Hence, to ensure that the grid spans the full boundary layer, it is necessary to multiply the flat plate distribution by the factor f_u which assumes a value between 1.5 and 5 depending on the angle of attack. It should also be noted that the variable $\tilde{s}$ is defined as the arc length, measured from the upstream stagnation point, along the airfoil surface or wake centerline. In the wake region, the value of $\tilde{s}$ gets larger than is intended with the formula of equation (2-9). Use of the square root of $\tilde{s}$ results in a more realistic approximation to the growth of the wake.

The first grid point above the surface is set at a constant distance $\tilde{n}_{\min}$ in the range 1×10^{-5} to 1×10^{-7}. This number is based on the assumption of unit chord. From experience gained with Navier-Stokes codes, it is known that locating the first point in this range results in computed value of n^+ near or less than one. In the present boundary layer code, the user specifies the distance of the first grid point above the airfoil surface. This value remains constant for all streamwise stations. Since the upper limit at each station grows as one moves downstream, while the number of points across the boundary layer remains constant, the stretching factor of equation (2-11) to locate the KBL-3 points between $\tilde{n}_2$ and $\tilde{n}_{KBL}$ is recomputed at each station. The resulting values are used to interpolate the $(\tilde{x},\tilde{y})$ coordinates of the boundary layer grid from the Navier-Stokes grid.

Special treatment of the stagnation point is required because equation (2-9) is singular when $\tilde{s}$ is zero. The upper grid limit is then computed from a linear extrapolation of the neighbouring points. Figure 6 shows the result. The grid lines obtained near the trailing edge are shown in figure 7.

Grid generation schemes other than the one above were studied, but were not found to be as successful. For example, computing the upper boundary distance by using the Navier-Stokes data to estimate local boundary layer thickness does not work well because there is no clear distinction between the boundary layer and the inviscid flow region. An attempt to compute the location of the first grid point above the surface using the wall shear stress was also made, but was not found to provide reliable results near the stagnation point and in regions of separated flow. It was therefore concluded that, although not optimal, the above grid generation scheme yields a smooth grid and does it efficiently.

After generating the Cartesian grid coordinates $(\tilde{x},\tilde{y})$, GBL transforms them to body conformal coordinates $(\tilde{s},\tilde{n})$ as required by the current boundary layer scheme. The $\tilde{s}$ coordinates are obtained from integration of the $\eta = 1$ lines. For any given ξ station, the

value of $\tilde{s}$ remains constant. The $\tilde{s} = 0$ coordinate is located at the leading edge stagnation point with the positive coordinates corresponding to the upper surface of the airfoil. The $\tilde{n}$ coordinates, defined to be zero on the airfoil surface or wake centerline, are obtained by integrating each ξ line from the $\eta = 0$ point.

The integration process to generate the body conformal coordinates is equivalent to unwrapping the Cartesian grid such that the $\eta = 1$ line is straight. It also means that η lines above the surface are compressed (their total arc length is reduced) in the process, but this is neglected because the boundary layer normal dimensions are very small when compared to the streamwise dimensions. A typical body conformal grid is shown in figure 8.

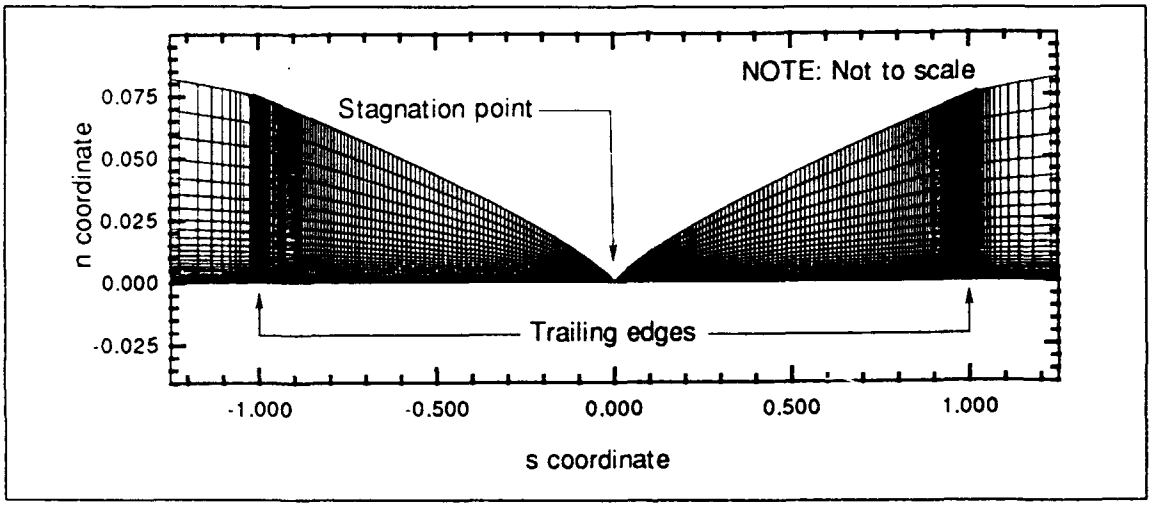

Figure 8. Boundary Layer grid in body conformal coordinates

COMPUTATIONAL TOOLS

Several computational tools are essential to the success of the boundary layer algorithm. For example, one tool is required to compute the metrics of the transformation when mapping the physical grid to the computational grid. Other tools transform the velocity components from one coordinate system to another, compute vorticity, or implement the Baldwin-Lomax turbulence model[11] (suitably modified for the body conformal coordinate system used in GBL). The main algorithm requires the solution of linear systems of equations at each streamwise station; a tridiagonal solver is used for the direct mode while the inverse mode requires a modified version of the same algorithm. Finally, other tools yield an estimate of the convergence of the algorithm.

Metrics of the Transformation

Use of the Cartesian $(\tilde{x},\tilde{y})$ or body conformal $(\tilde{s},\tilde{n})$ grid coordinates to solve the boundary layer equations increases the complexity of the finite difference algorithm because the spacing between points is not uniform. It is preferable to transform the equations to solve them on a uniform grid with unit spacing in both directions. This increases the complexity of the equations through the addition of metric terms, but simplifies the finite difference approximations to the derivatives and results in an overall gain in computational efficiency.

The physical grid points are mapped one-to-one to the computational grid with coordinates (ξ,η). The only exception, for the C-grid used in the present study, is the wake cut where each physical point is mapped to two computational points. Conversely, one can think of it as two computational points that overlap in the physical grid. Figure 9 shows an example of the physical (body conformal coordinates) and computational grids.

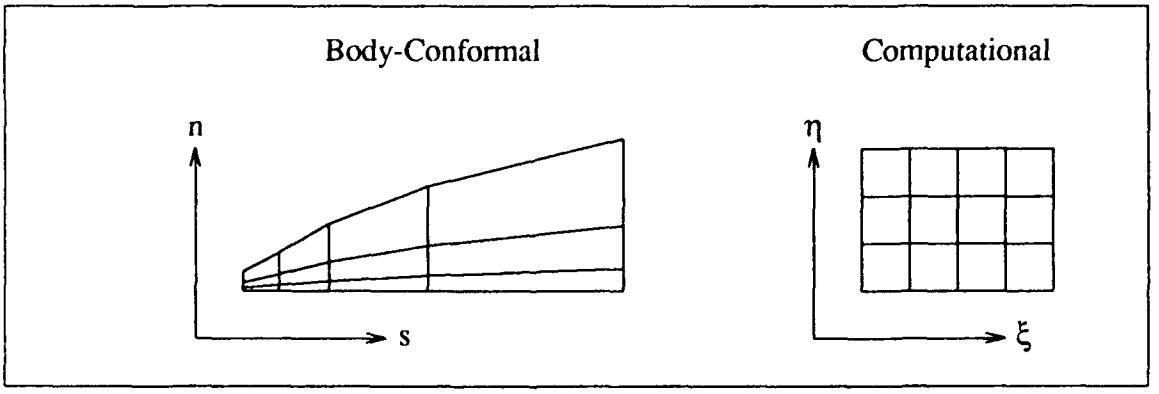

Figure 9. Body conformal and computational grids

As mentionned above, the transformed equations contain metrics of the transformation ($\xi_{\tilde{x}}$, $\xi_{\tilde{y}}$, $\eta_{\tilde{x}}$ and $\eta_{\tilde{y}}$) which must be determined numerically. This would normally involve

finite differences using the physical grid for which the ξ and η coordinates corresponding to each point are known. However, these computations use unequal grid spacing once again, exactly what we are trying to avoid. It is more practical to approach the problem from the other end. The physical coordinates corresponding to each (ξ,η) line intercept in the computational domain are known, so why not compute the derivatives $\tilde{x}_\xi$, $\tilde{x}_\eta$, $\tilde{y}_\xi$ and $\tilde{y}_\eta$ using the computational grid, and then compute the metrics from these? All that is required is the following relation linking the computational plane derivatives to the metrics.

The functional form of the computational variables is

$$\begin{aligned} \xi &= \xi(\tilde{x},\tilde{y}) \\ \eta &= \eta(\tilde{x},\tilde{y}) \end{aligned} \tag{3-1}$$

Using the chain rule, differential increments in the computational domain, $d\xi$ and $d\eta$, are linked to increments $d\tilde{x}$ and $d\tilde{y}$ in the physical space, i.e.

$$\begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \begin{bmatrix} \xi_{\tilde{x}} & \xi_{\tilde{y}} \\ \eta_{\tilde{x}} & \eta_{\tilde{y}} \end{bmatrix} \begin{bmatrix} d\tilde{x} \\ d\tilde{y} \end{bmatrix} \tag{3-2}$$

The role of the dependent and independent variables may also be reversed, i.e.

$$\begin{aligned} \tilde{x} &= \tilde{x}(\xi,\eta) \\ \tilde{y} &= \tilde{y}(\xi,\eta) \end{aligned} \tag{3-3}$$

for which the differential increments relation is

$$\begin{bmatrix} d\tilde{x} \\ d\tilde{y} \end{bmatrix} = \begin{bmatrix} \tilde{x}_\xi & \tilde{x}_\eta \\ \tilde{y}_\xi & \tilde{y}_\eta \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \tag{3-4}$$

Solving this last equation for $d\xi$ and $d\eta$ in terms of $d\tilde{x}$ and $d\tilde{y}$, and equating with equation (3-2), results in the relation

$$\begin{bmatrix} \xi_{\tilde{x}} & \xi_{\tilde{y}} \\ \eta_{\tilde{x}} & \eta_{\tilde{y}} \end{bmatrix} = \begin{bmatrix} \tilde{x}_{\xi} & \tilde{x}_{\eta} \\ \tilde{y}_{\xi} & \tilde{y}_{\eta} \end{bmatrix}^{-1} \tag{3-5}$$

which, in its expanded form gives the metrics

$$\begin{aligned} \xi_{\tilde{x}} &= J^{-1}\tilde{y}_{\eta} \quad , \quad \eta_{\tilde{x}} = -J^{-1}\tilde{y}_{\xi} \\ \xi_{\tilde{y}} &= -J^{-1}\tilde{x}_{\eta} \quad , \quad \eta_{\tilde{y}} = J^{-1}\tilde{x}_{\xi} \end{aligned} \tag{3-6}$$

where J, the Jacobian of the transformation, is defined as

$$J = (\tilde{y}_{\eta}\tilde{x}_{\xi} - \tilde{y}_{\xi}\tilde{x}_{\eta})$$

Cartesian to Body Conformal Velocity Transformation

During the initialization process for airfoil computations, GBL reads the ARC2D file inputs. These are in Cartesian coordinates, while GBL uses body conformal coordinates for its computations. Scalar quantities, such as state and energy variables, are independent of the coordinate system used, but vector quantities (such as velocity or momentum) are not and must be transformed from one coordinate system to another. The transformation requires the cosine angles relating the two coordinate systems. They are defined from the assumption that during grid generation, points at a given ξ station were located on the normal to the airfoil surface (see Figure 10), with the minor exception of a few lines in the trailing edge region as discussed previously.

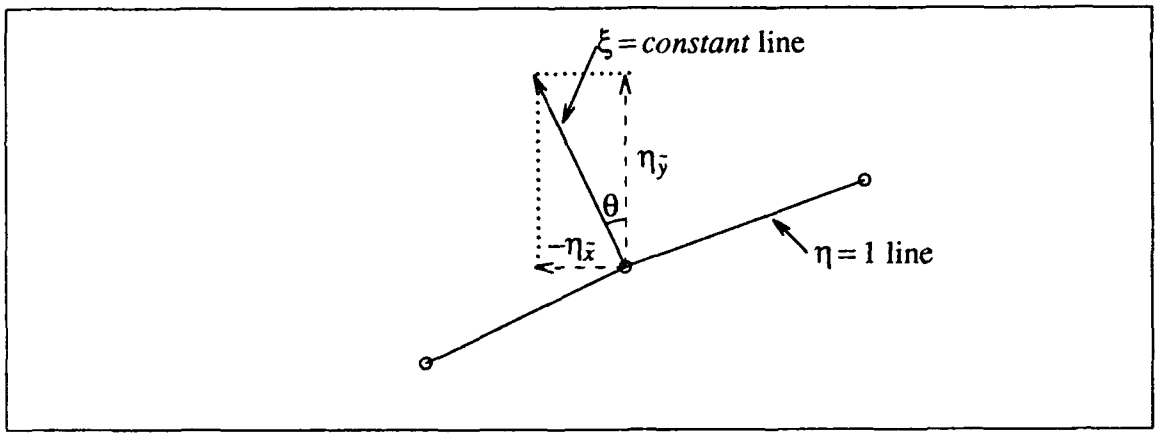

Figure 10. Rotation angle to transform velocity components

The metrics of the transformation $\eta_{\tilde{x}}$ and $\eta_{\tilde{y}}$, mapping the Cartesian grid coordinates to the computational domain, are then the $\tilde{x}$ and $\tilde{y}$ components of the vector η normal to the airfoil surface. The angle θ represents the rotation angle between the Cartesian and body conformal coordinate systems. Using the metrics, the sine and cosine of θ are defined as

$$\cos\theta = \frac{\eta_{\tilde{y}}}{(\eta_x^2 + \eta_y^2)^{1/2}}$$
$$\sin\theta = \frac{-\eta_{\tilde{x}}}{(\eta_x^2 + \eta_y^2)^{1/2}} \tag{3-7}$$

These trigonometric quantities relate the Cartesian and body conformal components of the velocity vector $\vec{V}$ through the equations

$$\tilde{u}_b = \tilde{u}_c \cos\theta + \tilde{v}_c \sin\theta$$
$$\tilde{v}_b = -\tilde{u}_c \sin\theta + \tilde{v}_c \cos\theta \tag{3-8}$$

The inverse relations are

$$\tilde{u}_c = \tilde{u}_b \cos\theta - \tilde{v}_b \sin\theta$$
$$\tilde{v}_c = \tilde{u}_b \sin\theta + \tilde{v}_b \cos\theta \tag{3-9}$$

The transformation sine and cosine angles are computed during initialization and stored in arrays SINANG() and COSANG(). The transformation from Cartesian to body conformal coordinates is handled by subroutine UVTC2B while the inverse transform, used during output, is done by subroutine UVTB2C.

Vorticity Computations

During initialization of airfoil cases, the vorticity is used to interpolate the ARC2D data in the η direction, for each ξ station, to find the location where the magnitude of the local vorticity $\tilde{\omega}$ is less than or equal to the user supplied cutoff value $|\tilde{\omega}_{cut}|$. The vorticity is defined in terms of Cartesian velocity components. In its non-dimensionalized form, it is

$$\tilde{\omega} = \frac{\partial \tilde{v}}{\partial \tilde{x}} - \frac{\partial \tilde{u}}{\partial \tilde{y}} \tag{3-10}$$

The metrics of the transformation from Cartesian to computational coordinates are used to replace the $\tilde{x}$ and $\tilde{y}$ derivatives and result in the expression

$$\tilde{\omega} = 0.5\left(\xi_{\tilde{x}}\tilde{v}_{\xi} + \eta_{\tilde{x}}\tilde{v}_{\eta} - \xi_{\tilde{y}}\tilde{u}_{\xi} - \eta_{\tilde{y}}\tilde{u}_{\eta} \right) \tag{3-11}$$

which is evaluated with centered differences wherever possible. At the boundaries, backward or forward three point differences are used.

Turbulence Modelling

The turbulent flow equations are derived from the laminar flow equations by replacing velocity, pressure, density and temperature by the sum of their mean and fluctuating components, i.e. $\tilde{u} = \overline{u} + u'$, $\tilde{p} = \overline{p} + p'$, etc. A time average of the equations is then taken and negligible terms are dropped. The resulting equations are identical to those for laminar flow except for additional terms such as the Reynolds stresses $-\rho\overline{u'v'}$. These express the increased diffusion characteristics of turbulent flow.

The computational algorithm of GBL applies to laminar as well as turbulent flow provided the increased diffusion is considered. For the momentum equation, the viscosity $\tilde{\mu}$ is replaced by the sum of the molecular viscosity $\tilde{\mu}_m$, obtained from Sutherland's formula, and a turbulent viscosity $\tilde{\mu}_t$ which replaces the Reynolds stresses. In the energy equation, the ratio $\tilde{\mu}/Pr$ is replaced by $\tilde{\mu}_m/Pr_m + \tilde{\mu}_t/Pr_t$ where $Pr_m = 0.72$ and $Pr_t = 0.9$ are the laminar and turbulent Prandtl numbers, respectively. A turbulence model is required to evaluate $\tilde{\mu}_t$.

The Baldwin and Lomax[11] model is widely used with the thin-layer Navier-Stokes equations. It uses a two layer algebraic eddy viscosity model to evaluate $\tilde{\mu}_t$. In the inner region, near a solid wall, it uses the Prandtl-Van Driest formulation. In the outer region, and in wakes, it uses the Clauser formulation. Transition from one formulation to the other is done at a crossover point, above the wall, defined as the first point where the two formulations are equal. The reader is referred to the original paper for the development and justification of the method.

The original Baldwin-Lomax equations are expressed below in terms of the scaled variables used in GBL. The inner region formulation is given by

$$\tilde{\mu}_{t_i} = \mathrm{Re}_{\infty} \cdot \tilde{\rho} \cdot \tilde{d}^2 \cdot |\tilde{\omega}| \tag{3-12a}$$

The variable $\tilde{d}$ is a characteristic length scale of the turbulence defined as

$$\tilde{d} = 0.4\,\tilde{n}\,(1 - e^{-n^+/26}) \tag{3-12b}$$

where n^+ is a local Reynolds number defined as

$$n^+ = \tilde{n}\left[\mathrm{Re}_\infty \frac{\tilde{\rho}_w \tilde{\tau}_w}{\tilde{\mu}_w^2}\right]^{1/2} \tag{3-12c}$$

In the outer region, the scaled turbulent viscosity is given by

$$\tilde{\mu}_{t_o} = 0.0168 \cdot 1.6 \cdot \tilde{\rho} \cdot \tilde{F}_{wake} \cdot \tilde{F}_{Kleb} \tag{3-13a}$$

The function $\tilde{F}_{Kleb}(\tilde{n})$ is the Klebanoff intermittency factor defined as

$$\tilde{F}_{Kleb} = \left[1 + 5.5\left[\frac{0.3\,\tilde{n}}{\tilde{n}_{\max}}\right]^6\right]^{-1} \tag{3-13b}$$

while the function $\tilde{F}_{wake}$ is defined as

$$\tilde{F}_{wake} = MIN\left\{\tilde{n}_{\max}\tilde{F}_{\max},\ \frac{\tilde{n}_{\max}\tilde{u}_{dif}^2}{\tilde{F}_{\max}}\right\} \tag{3-13c}$$

The quantities $\tilde{n}_{\max}$ and $\tilde{F}_{\max}$ are determined from the function

$$\tilde{F}(\tilde{n}) = \tilde{n}\,|\tilde{\omega}|\,(1 - e^{-n^+/26}) \tag{3-13d}$$

and the quantity $\tilde{u}_{dif}$ is the difference between the maximum and minimum total velocity in the profile, i.e.

$$\tilde{u}_{dif} = (\tilde{u}^2 + \tilde{v}^2)_{\max}^{1/2} - (\tilde{u}^2 + \tilde{v}^2)_{\min}^{1/2} \tag{3-13e}$$

Interpolation, using a local second order polynomial fit, improves the values of $\tilde{n}_{max}$ and $\tilde{F}_{max}$. In particular, the variation of $\tilde{F}_{max}$ is smoother in going from one station to the next.

The above model assumes Cartesian coordinates. It is modified for use with body conformal coordinates. First, the $\tilde{u}_c$ and $\tilde{v}_c$ components of velocity are replaced by their body conformal counterparts $\tilde{u}_b$ and $\tilde{v}_b$. Furthermore, outside the boundary layer edge defined by the index NDT, $\tilde{v}_b$ is neglected because it is unbounded outside this level, as explained before. This is particularly true near the leading edge region where the $\tilde{u}_b$ velocity gradient is very strong and the boundary layer is thin.

The second change is the replacement of the vorticity by the normal gradient of $\tilde{u}_b$, i.e. $|\tilde{\omega}| \rightarrow |\partial\tilde{u} / \partial\tilde{n}|$ The behavior of these two quantities is similar with differences in the magnitude of the terms, but use of the body conformal components compensates for that.

Direct and Inverse Solvers

The direct mode of GBL uses a finite difference scheme with centered or three point forward/backward difference stencils. The equations are solved at each ξ station, which results in a scalar tridiagonal systems of equations in the η direction, i.e.

$$[A]\vec{X} = \vec{B}$$

where $[A]$ is a tridiagonal matrix, $\vec{X}$ is the vector of unknowns to be solved for, and $\vec{B}$ is the right hand side vector, a known quantity. The efficient Thomas algorithm[12] is used to establish an upper triangular matrix followed by back substitution to yield a solution. Subroutine THOMAS, presented in Appendix B, solves the above system of equations.

For the inverse mode, the pressure forcing function, which is buried in the right hand side vector $\vec{B}$, must be replaced by an inverse function. GBL implements the inverse forcing functions of Van Dalsem and Steger[5]. They apply the momentum equation at the wall or on the wake centerline to derive an expression to replace the pressure. The resulting expression is a function of the wall shear stress over the airfoil, and of the wake centerline velocity in the wake. This yields a modified tridiagonal system of equations which has non-zero entries in the first column as shown below. A lower-upper decomposition scheme is used to solve this particular matrix equation; back substitution is applied to the upper triangular matrix to obtain an intermediate solution which is then used with the lower triangular matrix and forward substitution to yield the final result. The subroutine listing is included under the name INVTOM in Appendix B.

$$\begin{bmatrix} b_2 & c_2 & & & & \\ f_3 & b_3 & c_3 & & & \\ f_4 & a_4 & b_4 & \cdot & & \\ f_5 & & a_5 & \cdot\ \cdot & & \\ \cdot & & & \cdot\ \cdot & c_{n-3} & \\ \cdot & & & \cdot & b_{n-2} & c_{n-2} \\ f_{n-1} & & & & a_{n-1} & b_{n-1} \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \\ u_5 \\ \cdot \\ \cdot \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} d_2 \\ d_3 \\ d_4 \\ d_5 \\ \cdot \\ \cdot \\ d_{n-1} \end{bmatrix}$$

Convergence Estimates

It is useful to have an estimate of the convergence of the numerical algorithm. GBL implements two methods to produce such an estimate. The first method takes a simple difference of representative field variables from one cycle to the next, i.e. variable at time n minus variable at time $n-1$. The quantity used for each equation is: $\tilde{u}$ for momentum, $\tilde{H}$ for energy, and $\tilde{\rho}$ for continuity. The process involves all points of the field. The location and magnitude of the largest change is saved and the root mean square for the entire field is computed. Its logarithm (base ten) is presented, i.e.

$$\overline{\Delta Var} = \log_{10}(RMS) \tag{3-14}$$

The second method substitutes the variables back into their finite difference equations after each cycle. As the algorithm converges, the difference between the left and right hand sides of the equations approaches machine zero. Only interior points are used and equations (3-14) and (3-15) remain valid. Finally, when interpreting results obtained from either method, the user must be aware that convergence depends on the magnitude of $\Delta\tilde{t}$, the time step used to relax the equations.

INITIALIZATION OF VARIABLES

The numerical method used in GBL requires initialization of the forcing function variables ($\tilde{p}_e$, $\tilde{\tau}_w$ and $\tilde{u}_{wc}$), boundary condition variables ($\tilde{u}_e$ and H_e) and field variables ($\tilde{u}$, $\tilde{v}$, U, V, H, $\tilde{\rho}$ and $\tilde{\mu}$). The algorithm uses these initial values to estimate flow field derivatives and start the iterative cycle. Furthermore, because of to the non-linear nature of the boundary layer equations, the initial estimates must be representative of the flow field being computed; otherwise, the method may not converge. This type of behavior is characteristic of non-linear systems.

GBL supports four initialization procedures, two for test purposes and two for airfoil computations. The test cases are for laminar flow only. They are Falkner-Skan and Klineberg flows. The Falkner-Skan case is used to check the direct mode portion of GBL. The program uses theoretical distributions of pressure and velocity to initialize the flow field. The Klineberg case is used to exercise the inverse mode portion of GBL. It is a linearly retarded, mildly separated flow for which the distribution of wall shear stress is provided. The flow field is initialized with scaled Blasius profiles. It should be noted that although both cases are valid for laminar flow only; the Falkner-Skan test case can also be used to check the implementation of the turbulence model. All test case use the flat plate grid generated internally by GBL (as described in section 2).

The initialization process for airfoil flow is compatible with the ARC2D Navier-Stokes code. The procedure uses the ARC2D state vector and the corresponding grid to generate an "adaptive" grid or uses a subset of the Navier-Stokes grid. In the former case, the ARC2D data is interpolated linearly to initialize field values at the boundary layer grid points. For the latter case, the ARC2D field data is used directly. A coordinate transformation is then applied to the velocity data.

Velocity-Pressure Relation at Boundary Layer Edge

The Falkner-Skan test case has $\tilde{u}_e(\tilde{s})$ prescribed, but the corresponding edge pressure distribution, $\tilde{p}_e(\tilde{s})$, is unknown. It is the opposite for the airfoil initialization cases. The surface pressure from the Navier-Stokes solution may be used as $\tilde{p}_e(\tilde{s})$ if it is assumed constant across the boundary layer. However, finding $\tilde{u}_e(\tilde{s})$ is difficult because the Navier-Stokes solution does not show a clear transition between the boundary layer and the inviscid flow field. A relationship between $\tilde{u}_e(\tilde{s})$ and $\tilde{p}_e(\tilde{s})$ is then required.

The boundary layer effectively displaces the streamlines of the inviscid flow away from the solid surface by a distance equal to the displacement thickness of the boundary layer. If this fact is combined with the observation that the pressure remains constant across the boundary layer, Euler's inviscid momentum equation, which is valid along a streamline, may then be used to relate the edge velocity distribution $\tilde{u}_e(\tilde{s})$ to the surface pressure distribution $\tilde{p}_e(\tilde{s})$.

$$\frac{u_e^2}{2} + \frac{\gamma}{\gamma - 1}\frac{p_e}{\rho_e} = constant \tag{4-1}$$

The above *constant* is computed by replacing the edge values of velocity, pressure and density by the freestream equivalents. It assumes the flow is isentropic, i.e. it satisfies the relation

$$\rho = (constant)p^{1/\gamma} \tag{4-2}$$

which holds true for most subsonic and transonic flows under consideration in this study. Equation (4-2) is used to eliminate ρ_e in favor of p_e in equation (4-1). Scaling the variables in the resulting equation, we obtain expressions linking pressure and velocity at the boundary layer edge.

$$\tilde{p}_e = (\rho a_\infty^2)^{-1}\left[\frac{\rho_\infty}{p_\infty^{1/\gamma}}\left[\frac{u_\infty^2 - (a_\infty \tilde{u}_e)^2}{2}\left[\frac{\gamma - 1}{\gamma}\right] + \frac{p_\infty}{\rho_\infty}\right]\right]^{\frac{\gamma}{\gamma-1}} \tag{4-3}$$

$$\tilde{u}_e = \left[M_\infty^2 + \frac{2\gamma}{(\gamma - 1)\rho_\infty a_\infty^2}\left[p_\infty - p_\infty^{1/\gamma}(\rho_\infty a_\infty^2 \tilde{p}_e)^{\frac{\gamma-1}{\gamma}}\right]\right]^{1/2} \tag{4-4}$$

An expressions to compute the temperature at the boundary layer edge is obtained by combining the isentropic relationship with the perfect gas law

$$\tilde{T}_e = T_\infty\left[\frac{\tilde{p}_e}{p_\infty}\right]^{\frac{\gamma-1}{\gamma}} \tag{4-5}$$

The temperature is then used to compute total enthalpy at the boundary layer edge

$$\tilde{H}_e = \frac{c_p T_\infty}{a_\infty^2}\left[\frac{\tilde{p}_e \rho_\infty a_\infty^2}{p_\infty}\right]^{\frac{\gamma-1}{\gamma}} + \frac{\tilde{u}_e^2}{2} \tag{4-6}$$

Falkner-Skan Test Case

A flow field is called self-similar when velocity profiles $\tilde{u}(\tilde{s},\tilde{n})$ at different stations $\tilde{s}$ differ only by scale factors in $\tilde{u}$ and $\tilde{n}$. Schlichting[10] (pp.152-156) states the conditions under which self-similar conditions exist. One class of self-similar flow, studied extensively by Falkner and Skan[13], is obtained by specifying the boundary layer edge velocity as

$$u_e(s) = Cs^m, \qquad m = \frac{\beta}{2-\beta} \tag{4-7}$$

This function describes the potential flow in the neighbourhood of the stagnation point of a wedge whose included angle is $\pi\beta$. The distance s along the surface, is defined to be zero at the stagnation point. A non-dimensional form of this equation, with the constant C set equal to u_∞, is used in GBL.

$$\tilde{u}_e = M_\infty(\tilde{s}/l)^m \tag{4-8}$$

With the condition of equation (4-7), the continuity and streamwise momentum equations reduce to a single ordinary differential equation in terms of the stream function f and its derivatives with respect to a non-dimensional normal distance ζ.

$$f''' + \alpha f f'' + \beta(1 - f'^2) = 0 \tag{4-9}$$

subject to the boundary conditions

$$\zeta = 0:\ f = f' = 0 \quad ; \quad \zeta = \infty:\ f' = 1 \tag{4-10}$$

where $f' = \tilde{u} / \tilde{u}_e$

By carefully selecting values of the constants α and β, equation (4-9) approximates important existing flows. The constant α is often set equal to unity; this is the only case supported by GBL. If the constant β is set to zero, the equation describes flow over a flat plate at zero incidence. The case $\beta = 1$ describes the flow in the viscinity of a two dimensional stagnation point. Selecting any other values of β between zero and one, the equation describes the accelerating flow in the neighbourhood of the stagnation point of a wedge. Finally, by selecting small negative values of β, the equation represents decelarating flow; separation is encountered at $\beta = -0.199$. For values of $\beta < -0.199$, solutions of equation (4-9) are no longer unique; they are not attempted in GBL.

The user specifies the self-similar parameter m in the input file (see section 2). GBL uses this value to integrate equation (4-9) from $\zeta = 0$ to $\zeta = 10$ (a useful approximation to $\zeta = \infty$) with a Runge-Kutta scheme. The integration is sensitive to the second derivative at the wall, f''_w, and requires a shooting technique to update this value until convergence is achieved. The integration scheme yields distributions for the stream function $f(\zeta)$ and its derivatives $f'(\zeta)$ and f''_w.

Subroutine INIT0 initializes the flow field for Falkner-Skan test cases. The procedure starts by generating the Cartesian coordinates of the boundary layer grid, which, for this case, are also the body conformal grid coordinates. The metrics of the transformation are then computed and equation (4-9) is solved to obtain the Falkner-Skan solution requested by the user.

The boundary conditions and forcing functions are then computed. The edge velocity distribution $\tilde{u}_e(\tilde{s})$ is obtained from equation (4-8) while equation (4-3) is used to obtain the pressure $\tilde{p}_e(\tilde{s})$. Total enthalpy is computed from equation (4-6). The wall shear stress distribution $\tilde{\tau}_w(\tilde{s})$ is computed from the theoretical Falkner-Skan distribution, i.e.

$$\tilde{\tau}(\tilde{s}) = l\,\tilde{u}_e f''_w \left[\frac{m+1}{2} \frac{\tilde{u}_e a_\infty}{\nu_\infty |\tilde{s}|\, l} \right]^{1/2} \tag{4-11}$$

Finally, the wake centerline velocity, which is not used for this case, is set equal to zero. Having initialized the forcing function and boundary condition arrays, the velocity components u and v are computed for each grid point. Using the s and n values, the scaled value ζ is computed from

$$\zeta = \tilde{n} \left[\frac{m+1}{2} \frac{|\tilde{u}_e| a_\infty}{\nu_\infty |\tilde{s}|\, l} \right]^{1/2} \tag{4-12}$$

which is used to interpolate values of f and f'. These values are then used in the following expressions to compute $\tilde{u}$ and $\tilde{v}$

$$\tilde{v} = \tilde{u}_e f' \tag{4-13}$$

$$\tilde{v} = a_\infty^{-1} \left[f + \frac{m-1}{m+1} \zeta f' \right] \left[\frac{m+1}{2} \frac{\nu_\infty |\tilde{u}_e| a_\infty}{|\tilde{s}|\, l} \right]^{1/2} \tag{4-14}$$

The final steps of INIT0 are to initialize the contravariant velocity components, density and viscosity throughout the field. For turbulent flow, the turbulent viscosity is also computed.

Klineberg Test Case

Klineberg and Steger[14] used an inverse method to solve the laminar boundary layer equations for linearly decelerated flows over a flat plate. The wall shear stress distribution was specified as an analytic function to generate a small region of reversed flow. Their paper gives an example wall shear stress distributic

$$\begin{aligned} \bar{\tau} = \bar{\tau}_0 = \frac{0.33238}{12}(\tilde{s}-2)(\tilde{s}-6) \quad , \quad 0 < \tilde{s} < 2 \ , \ \tilde{s} > 6 \\ \bar{\tau} = \bar{\tau}_0(1+0.1(\tilde{s}-2)(\tilde{s}-6) \ , \ 2 \le \tilde{s} \le 6 \end{aligned} \tag{4-15}$$

and the corresponding numerical results.

Routine INIT1 initializes the flow field variables for the Klineberg case. The first operation is to generate the boundary layer grid and to compute the metrics of the transformation. The boundary conditions and forcing functions are then initialized. Klineberg's edge velocity is read from a file, rescaled using the freestream Mach number, and a spline is fit through the data to interpolate values corresponding to the boundary layer grid. Equation (4-3) is then used to compute the edge pressure and equation (4-6) for total enthalpy at the edge. The wall shear stress distribution of equation (4-15) is rescaled as follows for use in GBL

$$\tilde{\tau}_w = \bar{\tau}\tilde{u}_e \left[\frac{\mathrm{Re}_\infty \tilde{u}_e}{M_\infty \tilde{s}} \right]^{1/2} \tag{4-16}$$

Finally, the wake centerline velocity, which is not used, is set equal to zero.

Since the Klineberg case starts from flat plate flow, the field velocity components $\tilde{u}$ and $\tilde{v}$ are initialized using flat plate profiles generated from the Falkner-Skan equation with $\alpha = 1$ and $\beta = 0$. The remaining variables are then initialized. The contravariant velocity arrays are computed from the metrics of the transformation and the body conformal velocity components $\tilde{u}_b$ and $\tilde{v}_b$. Density is computed from the perfect gas relation, viscosity from Sutherland's law, and turbulent viscosity is set to zero.

Airfoil Cases

Two initialization procedures were designed for airfoil cases. INIT2 uses a subset of the Navier-Stokes grid as the boundary layer grid, while INIT3 generates an adaptive boundary layer grid as discussed in section 2. The two procedures are identical with the following exception: the Navier-Stokes data is used directly in INIT2, but the use of a different boundary layer grid in INIT3 requires interpolation of the Navier-Stokes data.

The initialization process starts with the input of the Navier-Stokes grid Cartesian coordinates and the corresponding state vector Q file which contains the non-dimensional density, x-momentum, y-momentum and total energy at each point, i.e.

$$Q = \begin{bmatrix} \tilde{\rho} \\ \tilde{\rho}\tilde{u}_c \\ \tilde{\rho}\tilde{v}_c \\ \tilde{e} \end{bmatrix} \tag{4-17}$$

The energy $\tilde{e}$ is related to total enthalpy by the relation

$$\tilde{H} = \frac{\gamma\tilde{e}}{\tilde{\rho}} - (\gamma - 1)\left[\frac{\tilde{u}_c^2 + \tilde{v}_c^2}{2}\right] \tag{4-18}$$

The subscript c on the momentum elements of Q denotes that Cartesian components of velocity are used. They are used with the metrics of the transformation to go from the Navier-Stokes to computational grids, and equation (3-11), to compute vorticity. The metrics are then used with equation (3-7) to compute the rotation angles necessary to transform the the velocity components to body conformal coordinates, and the velocity components are transformed.

The magnitude of vorticity is strongest near the airfoil surface and vanishes rapidly as we move out of the boundary layer. It is therefore used to estimate the location of the boundary layer edge. The user selects the cutoff vorticity level $|\tilde{\omega}_{cut}|$ which defines the edge and GBL interpolates the vorticity data to find this line. The corresponding edge pressure $\tilde{p}_e$ and velocity $\tilde{u}_e$ may also be used as boundary conditions if the user sets the input variables RSPEDG and RSUEDG equal to 1 in the input file. The user may also elect to use the surface pressure and the corresponding edge velocity computed from equation (4-4) as edge conditions. The INIT2 routine may also use the pressure and velocity at $\eta = KBL$ as boundary conditions; INIT3 does not offer this option.

The stagnation point is located at the ξ station with the highest pressure. In addition, the Cartesian grid data is searched to identify the ξ stations corresponding to the leading edge and lower/upper trailing edges. The origin of the body conformal grid is then moved to the stagnation point, so that $|\tilde{s}|$ becomes the arc length distance from the stagnation point, and the metrics of the transformation from body conformal to computational coordinates are computed. The contravariant velocity components are also computed at this stage.

The next steps for the initialization are to reset the total enthalpy if the constant enthalpy option is selected, compute the molecular and turbulent viscosities, and if the user elects, to reset the velocity profiles at the stagnation and neighbouring points to stagnation ($m = 1$) Falkner-Skan profiles. This last option is included in GBL to study the impact of the upstream conditions on the computations.

Since the user selects the boundary layer edge arbitrarily with tne variable $|\tilde{\omega}_{cur}|$, additional grid points are added above the previously computed boundary layer edge to make sure that the grid spans the entire boundary layer. The wake centerline velocity is then set equal to the $\tilde{u}_b$ velocity along the line $\eta = 1$ and the edge total enthalpy is set to the enthalpy at the points $\eta = NDT(\xi)$. The wall shear stress forcing function is set to zero in the wake while, on the airfoil surface, it is computed using a three point forward difference of the velocity component tangential to the wall, i.e.

$$\tilde{\tau}_w = \eta_y \Delta_\eta \left[\frac{3 - E_\eta^{+1}}{2} \right] \tilde{u}_b \qquad (4\text{-}19)$$

[BLANK PAGE]

SOLUTION SCHEME

The momentum, energy and continuity equations share a common solution process. The computational field is swept in a ξ direction, and at each station, an implicit system of linear equations in the η direction is solved. GBL takes advantage of this commonality to establish three levels of subroutines for the solution process. The highest level controls the overall solution cycle by calling each equation solver in a particular order. The second level solves a given equation (e.g. momentum) over portions of, or over the entire computational field. Check functions and forcing function updates are also done at the second level. The third level evaluates specific functions for the equation solvers. The above structure is illustrated in Figure 11. It provides the flexibility required for the research framework of GBL. Changes to the formulation, boundary conditions or overall sweeping schemes are easily implemented by modifying the appropriate modules. Elements for each level are now discussed.

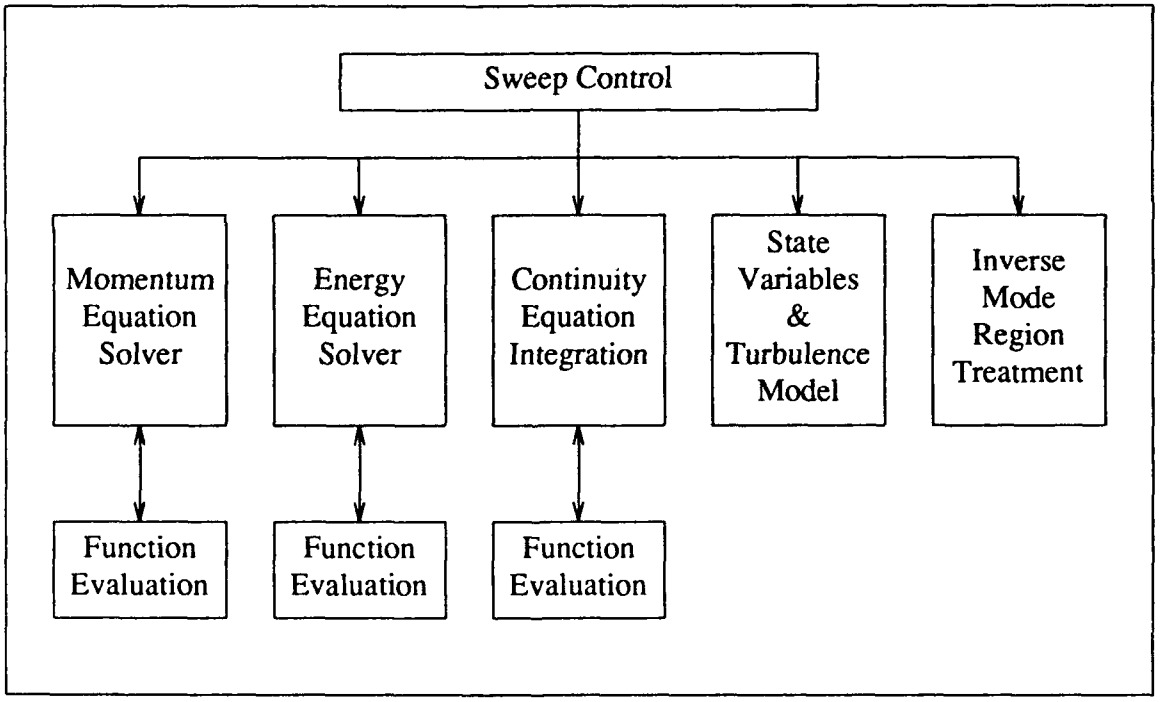

Figure 11. Levels in solution process

Sweep Control Level

The solution scheme developed by Van Dalsem and Steger involves the solution of the equations in a particular order over a user specified number of cycles. During each cycle, the solution progresses as follows:

[1] The streamwise velocity component $\tilde{u}$ is updated for a portion or the full computational field by solving the momentum equation. The other equations are solved for the same extent of the field.

[2] The contravariant components of velocity, $\tilde{U}$ and $\tilde{V}$, are updated using their definition and the latest values of $\tilde{u}$.

[3] Total enthalpy, $\tilde{H}$, is updated using the energy equation. Alternatively, the assumption of constant enthalpy may be used.

[4] Density, $\tilde{\rho}$, is updated using the latest values of total enthalpy and streamwise velocity.

[5] The normal velocity component $\tilde{v}$ is updated by integrating the continuity equation from the $\eta = 1$ line and using the latest values of the streamwise velocity components and density.

[6] The contravariant velocity component $\tilde{V}$ is once more updated using the latest value of $\tilde{v}$. $\tilde{U}$ does not depend on $\tilde{v}$ and needs no updating.

[7] Finally, the molecular viscosity is computed from Sutherland's law while the Baldwin-Lomax turbulence model is used to update turbulent viscosity.

It is not clear how the computational field should be swept. The parabolic nature of the boundary layer equations dictates that sweeps be done in a downstream direction, but how are embedded regions of reverse flow handled? The current sweeping schemes of GBL attempt to answer this question. Three sweep procedures are incorporated in the program. One routine, CMPGB0, is used with the flat plate grid to test algorithms. The user controls the direction and range of the sweeps. The other routines are used for airfoil cases.

Routine CMPGB1A sweeps a user selected range of the computational field, from one end to the other, in alternating directions from one cycle to the next. The decision to alternate sweep direction is based on observations of the behaviour of the algorithm for flat plate flows. First, the algorithm converges when sweeping the flow field in the general downstream direction, which is consistent with the parabolic nature of the equations. When sweeping in the opposite direction, the algorithm remains stable for two to five iterations and then diverges rapidly. The time step also affects the rate of divergence. Alternating the sweep direction may present the advantage that every second cycle, the parabolic nature of the equations is respected in regions of reversed flow. Hence, if convergence of the algorithm is slowed down because the reversed flow region is swept against the local flow direction, alternating the sweep direction may improve overall convergence. Furthermore, alternating the direction of sweep for the entire field simplifies the overall sweeping scheme.

The last sweeping scheme, CMPGB1D, uses the pressure distribution provided by ARC2D to break the computational field in two segments about the leading edge stagnation point. Separate solutions are obtained for the lower and upper surface segments with each segment being swept in its general downstream direction. The two upstream stations, the stagnation and adjacent stations, are fixed to the Navier-Stokes data. They are not updated, but still influence the solution since they are the upstream boundary conditions in ξ.

It should be noted that the current approach treats the lower and upper wake separately. This may lead to a discontinuity of the flow variables at the wake centerline. This is not considered a problem for the present study and could be corrected in later versions of GBL. It would involve use of points on opposite sides of the computational domain to ensure continuity at the wake centerline. Such treatment is not included here.

Equation Level

Solution of individual equations or the update of state variables and the turbulence model are done at the equation level. GBL also includes a scheme to update the inverse forcing function $\tilde{\tau}_w$ to match pressure at the edge. Recall from section 3 that, for isentropic flows, pressure and velocity are related at the edge. The current update scheme matches $\tilde{u}_e$ in the reverse flow region. The five parts of the equation level are now described separately.

Momentum equation

The sweep control level sets the range [MINXI,MAXXI] and sweep direction RSWEEP for the solution of the momentum equation. The actual solution is done in routine USWEES. For each station, the following sequence of operations is required. First, the velocity $\tilde{u}^n$ is saved to evaluate the time derivative $\tilde{u}_t$ and to compute the residuals. For airfoil cases, the next operation is to set the switch LSMODE to *true* or *false*. Selection of the mode for each station is actually done as part of the inverse mode region update which are discussed below.

The direct and inverse mode computations are handled by separate blocks of routine USWEES. Each block evaluates the matrix elements for the tridiagonal or modified tridiagonal matrix solvers, sets the boundary conditions, and solves the system of equations. The user has partial control over the boundary conditions. Over a solid boundary, Dirichlet or zero gradient Neumann (slope equal zero) boundary conditions may be applied at the lower and upper η limits of the profile. In the wake, a zero gradient Neumann condition is imposed at both boundaries irrespective of the mode. The user controls the boundary conditions through variables BCDXL and BCDXU for the direct mode, and through variables BCIXL and BCIXU for the inverse mode.

The matrix solver returns the solution at time $n+1$ in the vector DT. Under or over-relaxation is then applied to the solution depending on the factor WX. This factor is added mostly for convenience and is set to unity for most cases. Note that by updating the values of $\tilde{u}$ after each station, this data influences the computations at the next station.

Energy equation

Solution of the energy equation is similar to that for the momentum equation. Routine HSWEES solves the equation on a per station basis for the range and direction set at the sweep control level. The first operation is to save the solution at time n. The implicit and forcing term coefficients are then computed, the boundary conditions applied, and the resulting tridiagonal system of equations solved. The user selects between Dirichlet and zero gradient Neumann conditions at the lower and upper boundaries through variables BCDEL and BCDEU respectively. This selection applies to all ξ stations including those in the wake. The solution may be under or over-relaxed by setting the variable WH to a value other than unity.

Continuity equation

The continuity equation simply relates the components of velocity and density to ensure that new mass is not created. Using the $\tilde{u}$ components of velocity and density obtained from previous steps of the solution cycle, the $\tilde{v}$ components of velocity are obtained with an explicit integration process. The value of $\tilde{v}$ is required at one location to start the integration. In the present case, it is known to be zero on the body surface, due to the no slip condition, and it is also zero on the wake centerline by definition. The integration process is carried out in module CINTEG over the range [MINXI,MAXXI]. Direction of the sweep has no effect.

Although the integration is explicit, the tridiagonal implicit solver is used to solve each profile. However, only the lower and main diagonals are used, which yields a bidiagonal system of equations. This means that matrix elements are computed for the upper boundary of the profile while the Dirichlet boundary condition $\tilde{\rho}\tilde{v} = 0$ is applied at the lower boundary. Solution of the resulting matrix, using the tridiagonal solver, yields values of $\tilde{\rho}\tilde{v}$ for the profile. The values of $\tilde{v}$ are obtained by dividing the matrix solution by the local density. Values of the contravariant velocity V are also updated at the same time. Note that the other contravariant velocity component, $\tilde{U}$, does not depend on $\tilde{v}$ and need not be updated.

State variables & turbulence model

The state variables, $\tilde{\rho}$ and $\tilde{\mu}_m$, are properties of the flow which depend only on the local level of energy, i.e. $\tilde{H}$, $\tilde{p}$, and the velocity components. Their computation is done on a

point-by-point basis within the range [MINXI,MAXXI] and without need to sweep in a particular direction. The perfect gas law is used in routine UPRHO to compute density. Routine UPRMM uses Sutherland's approximation, with reference temperature T_∞ to compute molecular viscosity.

The Baldwin-Lomax turbulence model is used in routine UPRMT to compute turbulent viscosity. This process involves all points in a given profile because the maximum velocity differential must be computed. Only stations in the range [MINXI,MAXXI] are updated.

Inverse mode region treatment

Routine SCHECK is an algorithm to detect regions of reversed flow on the airfoil. The airfoil surface is scanned from the lower trailing edge to the upper trailing edge, omitting the forward stagnation point (i.e. a zone cannot include the leading edge stagnation point). For each scan, temporary pointers are set, based on the velocity gradient at the wall, to mark the beginning and end of up to four zones of reversed flow. These are normally found immediately after a shock wave or near the upper trailing edge of a transonic airfoil at positive angle of attack. The location of the temporary pointers is then compared to that of permanent pointers and adjusted only if a zone expands. Allowing the reversed flow region to shrink results in oscillations and non-convergence of the code.

Having identified the regions of separation, routine UPTAUW updates the inverse forcing function $\tilde{\tau}_w$ if the user has enabled the pressure matching option (LSPMAT = *true*). This algorithm updates the wall shear stress according to a formula proposed by Van Dalsem and Steger[6]

$$\tilde{\tau}_w^{i+1} = \tilde{\tau}_w^i + |\tilde{\tau}_w^i| \, (\tilde{u}_e - \tilde{u}_{NDT}) \tag{5-1}$$

The inverse forcing function is not updated after every cycle because a change of $\tilde{\tau}_w$ often produces perceptible changes of the velocity profile over several iterations, and changing the forcing function too soon leads to instability of the algorithm. The logic to decide when to update $\tilde{\tau}_w$ is incorporated in routine UPFORF. It monitors the variation of $\tilde{u}_{NDT}$ from one cycle to the next, and updates $\tilde{\tau}_w$ only if the percentage change is less than a user supplied value UECHK, but not before three iterations have been completed. This last condition results from experience gained with GBL, i.e. that as a rough rule-of-thumb, at least three to five iterations are required before updating the forcing function. The problem with this method is that the tolerence UECHK must be fairly coarse during the initial cycles to allow more adjustments of $\tilde{\tau}_w$, but gradually tightened as convergence is achieved. To partially correct for this, the user stipulates the maximum

number of iterations, MAXCNT, beyond which the forcing function is updated. Once the value of $\tilde{u}_{NDT}$ has converged to $\tilde{u}_e$ within a user supplied tolerance UETOL, say ½%, routine UPTAUW is disabled.

Function Level

The matrix elements are evaluated at the function level, the most basic level of the current numerical method. Recall that the finite difference molecule yields a tridiagonal system of equations in η[8]. It results that at each point, the momentum, energy or continuity equation is conveniently expressed as

$$A_k\Theta_{j,k-1}^{n+1} + B_k\Theta_{j,k}^{n+1} + C_k\Theta_{j,k+1}^{n+1} = D_k \tag{5-2}$$

where Θ is $\tilde{u}$ for the momentum equation, $\tilde{H}$ for energy, or $\tilde{\rho}\tilde{v}$ for continuity. A, B and C are the coefficient of the implicit side while D is the forcing term. For the modified tridiagonal system of equations used to solve the inverse mode form of momentum, the above equation becomes

$$F_k\Theta_{j,2}^{n+1} + A_k\Theta_{j,k-1}^{n+1} + B_k\Theta_{j,k}^{n+1} + C_k\Theta_{j,k+1}^{n+1} = D_k \tag{5-3}$$

Basic functions are used to evaluate the implicit coefficients and the forcing term at a given ξ station. Since the formulation of each equation changes near the boundaries of the computational field, different basic functions are used at these locations. This is the case for the two ξ stations at each end of the flow field. Different forms of the implicit coefficient B and forcing term D are used for the direct and inverse mode of the momentum equation. Furthermore, the inverse mode functions for the momentum equation are different for body and wake stations.

Use of the basic functions makes the algorithm very flexible. Changes to the equations simply require new basic functions.

CODE TESTING

This section presents selected results to demonstrate the ability of the code to compute boundary layer flows. First, a flat plate grid geometry is used to verify coding accuracy for the direct and inverse mode algorithms. These tests involve laminar flow only. The implementation of the Baldwin-Lomax turbulence model is then verified by comparing flat plate computations against experimental results.

The second part of this section presents results from simple numerical experiments to explore the convergence characteristics of the code. The effect of the time-like variable and sweep direction are discussed. In the third part, the ability of the code to solve the equations for an airfoil under attached flow conditions is shown. In the last part, airfoil results for more challenging conditions, with embedded supersonic and reversed flow regions, are presented.

Code Verification

The implementation of the algorithms is verified by conducting numerical experiments and comparing the results to analytic solutions for the direct mode, and to other computational results for the inverse mode. Each test problem exercises one mode only. The flow is laminar and the grid is kept as simple as possible, a flat plate in this instance. Results from this type of problems are very informative with respect to issues as diverse as grid dependencies, convergence characteristics and accuracy. The test direct mode test case can also be used to verify the implementation of the turbulence model by computing a turbulent flat plate flow and comparing the results to experimental data. Results for the three test problems are now presented and discussed while the convergence issue is addressed later in the section.

Direct mode test

The direct mode algorithm is verified against laminar Falkner-Skan solutions because a wide range of pressure gradients, from separation to strongly accelerated flows, may be simulated. Falkner-Skan flows are also laminar which eliminates uncertainties associated with the turbulence model and its experimental roots. For the current case, freestream conditions are selected to yield incompressible laminar flow:

$$\mathrm{Re}_\infty = 116{,}500$$

$$M_\infty = 0.005$$

$$T_\infty = 288\,K$$

$$l = 1.0$$

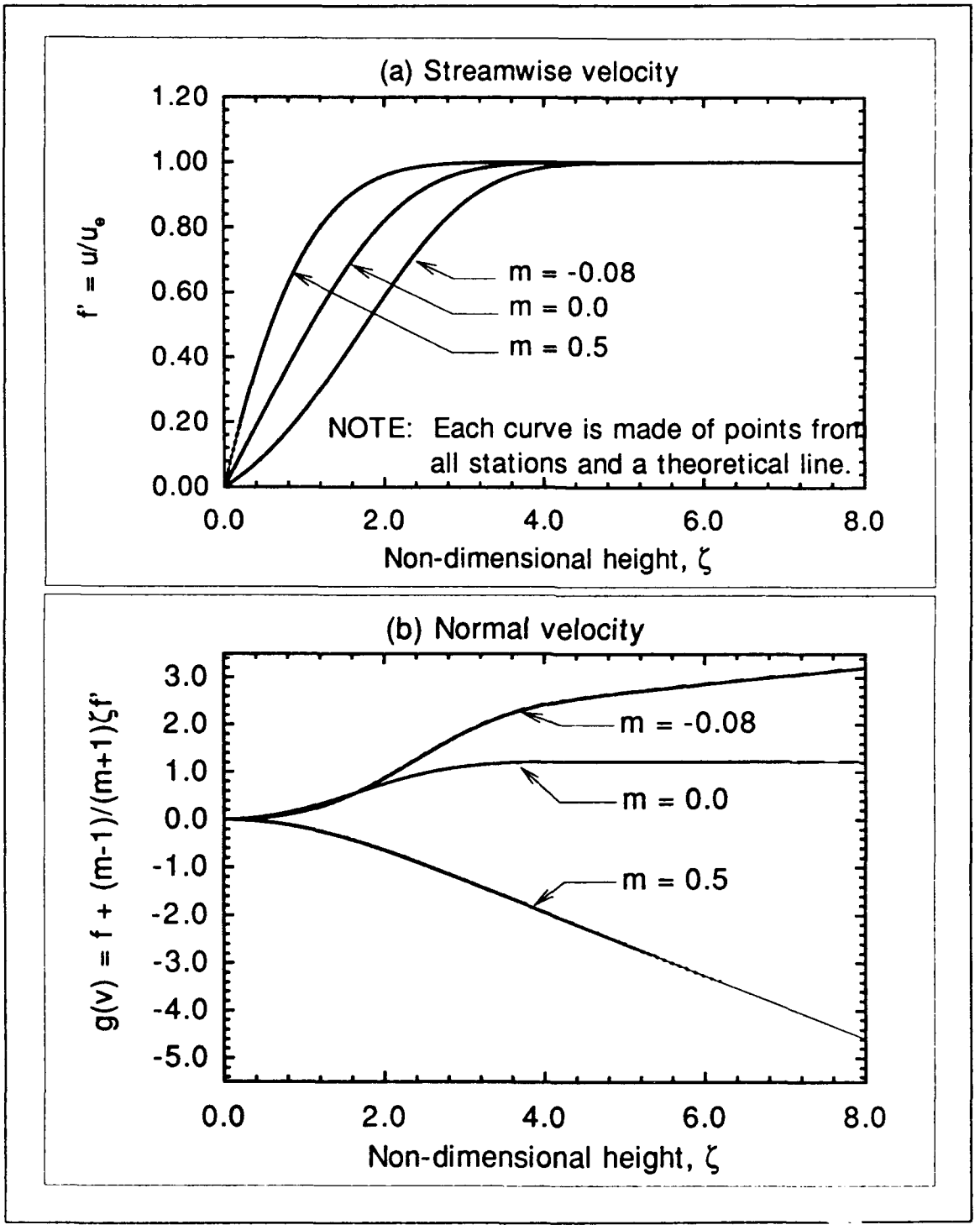

Figure 12. Falkner-Skan results for three flow types

and are kept constant for all Falkner-Skan computations. Similarly, a single grid with 90 streamwise stations and 50 points in the normal direction is used for all cases. It is generated internally using the parameters:

$$\tilde{s}_s = 0.01$$

$$\tilde{s}_e = 1.00$$

$$\Delta\tilde{s}_0 = 0.0001$$

$$f_l = 0.2$$

$$f_u = 2.5$$

which yield a streamwise expansion factor $\varepsilon_s = 0.0775$ and a maximum normal expansion factor $\varepsilon_n = 0.0866$ in equation (2-11). Note that the grid starts at $\tilde{s} = 0.01$ instead of $\tilde{s} = 0$ because equations (4-12) and (4-14) for the variables ζ and $\tilde{\upsilon}$, respectively, are singular at this last point. This complicates the initialization of the velocity profiles locally.

Three cases are computed for values of the self-similar parameter $m = -0.08$, $m = 0.0$ and $m = 0.5$. These correspond to flow under adverse pressure gradient, constant pressure flow (i.e. flat plate flow), and accelerating flow conditions, respectively. The theoretical solutions are used to initialize the flow field variables in each case. To verify that the numerical solution really converges to the theoretical solution, the initial data is corrupted by setting the velocity values $\tilde{u}$ and $\tilde{\upsilon}$ at all non-boundary points to 98% of the theoretical value. The boundary conditions and upstream conditions (at the first two stations) are set to their correct theoretical values. During the solution process, the time-like variable Δt is set to 100. Results for the three test cases are shown in Figure 12. Excellent agreement is obtained between the numerical results and theory which demonstrates the accuracy of the direct mode algorithm for laminar flow.

In achieving the above results, careful selection of the grid is necessary. For flows subject to a pressure gradient, it is important to concentrate sufficient streamwise stations to resolve the pressure terms; otherwise, the accuracy is degraded. This applies to stations near the leading edge where pressure gradients are strong, and near the trailing edge of the flat plate, where spacing of the streamwise stations is coarse. Furthermore, near the trailing edge, the coarseness of the grid even results in different values of the pressure terms if a three-point-backward finite differences is used instead of the centered difference.

The extent of the grid in the normal direction is also important. If the grid does not cover the full extent of the boundary layer, the boundary condition is imposed at a location where it does not apply and affects the shape of the local velocity profile. This problem is experienced with the flat plate grid generation scheme built into GBL. It is partially

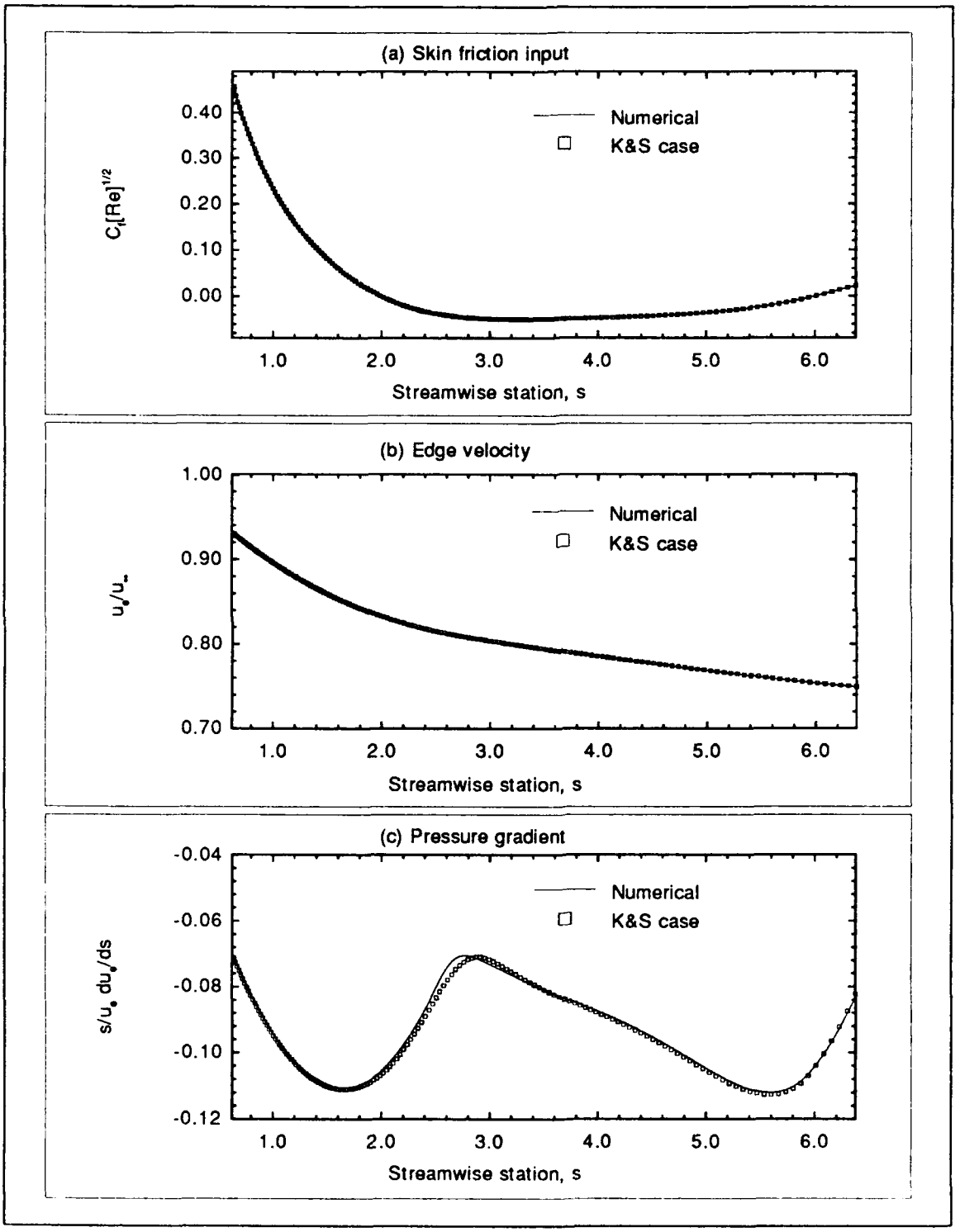

Figure 13. Klineberg flow results

eliminated with the use of the f_l and f_u multiplication factors, but the fact remains that the growth of the boundary layer grid is that for a constant pressure flow. GBL would benefit from an improved grid generation scheme, or a truly adaptive grid scheme which adapts itself to the flow conditions.

Inverse mode test

The implementation of the inverse mode algorithm is verified by computing the laminar flow over a flat plate subjected to the linearly decelerated flow of Klineberg. Freestream conditions are the same as those used for the Falkner-Skan test cases, but a different flat plate geometry must be used for compatibility with Klineberg's results. The grid generated has 200 streamwise stations and 60 points across the boundary layer. The parameters are:

$$\tilde{s}_s = 0.03$$

$$\tilde{s}_e = 6.375$$

$$\Delta\tilde{s}_0 = 0.03$$

$$f_l = 0.5$$

$$f_u = 2.0$$

The number of streamwise stations and the spacing are selected to yield approximately constant spacing in the $\tilde{s}$ direction, thus more stations are present in the reversed flow region. The flow field is initialized with constant pressure profiles ($m = 0$) scaled with Klineberg's edge velocity distribution.

Figure 13 compares the numerical results to those of Klineberg. The skin friction distribution used to drive the solution is shown in figure 13(a) while figures 13(b) and 13(c) show the corresponding edge velocity and pressure gradient results. The computed edge velocity matches the Klineberg results very closely, which is noteworthy since the momentum equation uses a Neumann boundary condition at the upper η edge. The pressure gradient, defined as

$$\overline{m} = \frac{\tilde{s}}{\tilde{u}_e} \frac{d\,\tilde{u}_e}{d\,\tilde{s}}$$

differs in the reversed flow region. However, comparison of the current results with those of Klineberg is difficult because the above pressure gradient derivative is computed from the numerical results. In particular, the evaluation of the term $d\,\tilde{u}_e / d\tilde{s}$ depends on the grid and the grid used in the current method is very different grid than the one used

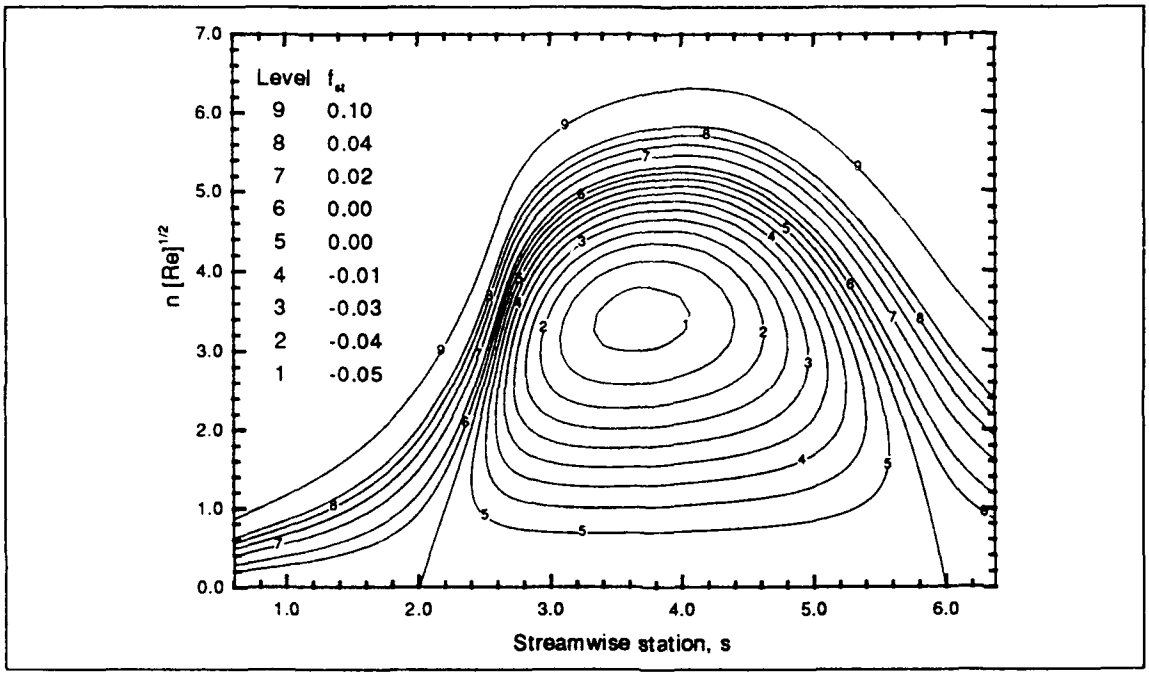

Figure 14. Streamlines for Klineberg flow

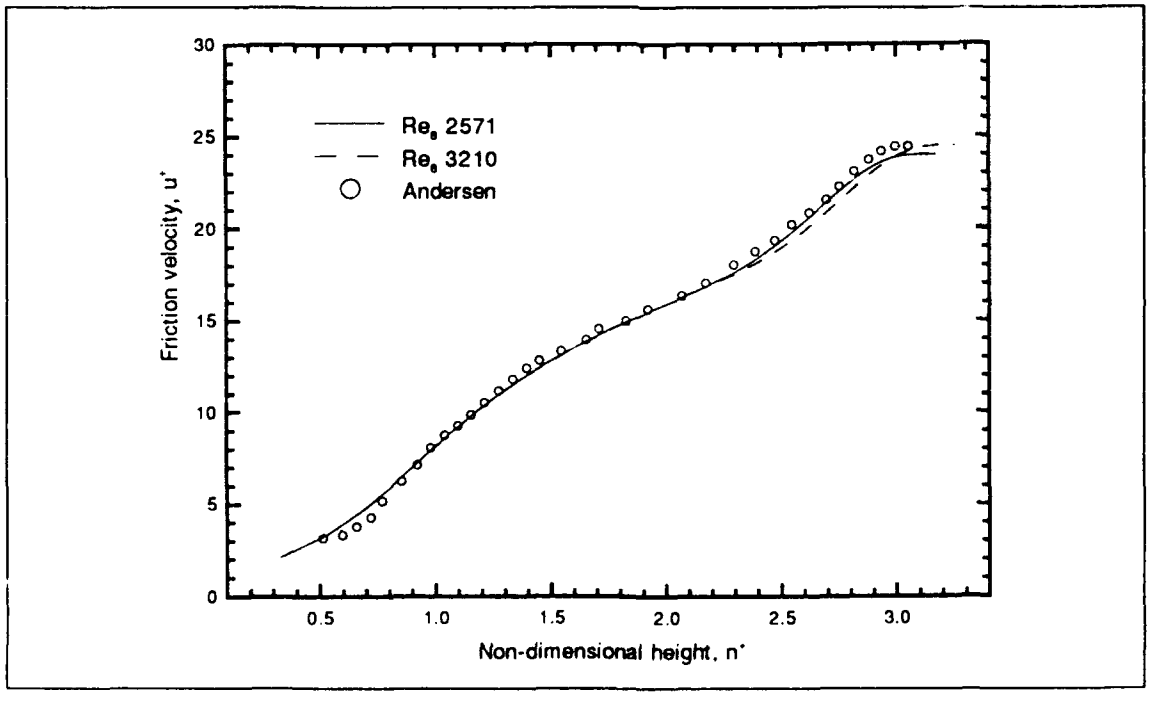

Figure 15. Turbulent flat plate flow - Andersen experiment

by Klineberg; a body conformal grid with clustering of the η lines near the wall is used here, while Klineberg uses a uniform grid with coarse spacing since his method exploits transformed variables instead of scaled variables.

Figure 14 presents the streamlines computed for the current computations. They correspond closely to the results of Klineberg and show that GBL captures the salient features of the flow.

Turbulence model test

A test of the turbulence model implementation is carried out by computing the flat plate flow measured by Andersen[15]. The experimental conditions for this flow are:

$$\mathrm{Re}_\infty = 6.5\times10^5$$
$$M_\infty = 0.028$$
$$T_\infty = 293\,K$$
$$l = 1.0$$

Andersen used a wind tunnel with a 2.44 meter long working section to simulate flat plate flow and measured the mean velocity profiles at various streamwise stations. In the current computations, the flat plate is approximated by a grid with 90 streamwise stations, 50 points across the boundary layer and the following parameters:

$$\tilde{s}_s = 0.01$$
$$\tilde{s}_e = 2.44$$
$$\Delta\tilde{s}_0 = 0.005$$
$$f_l = 0.5$$
$$f_u = 1.3$$

The flow field is initialized with the laminar flow variables. The momentum equation is solved with a Dirichlet condition at the upper boundary in η. Andersen presents his results in boundary layer coordinates u^+ and n^+ which are defined as

$$u^+ = \frac{u}{u_\tau} \quad , \quad u_\tau = \left[\frac{\tau_w}{\rho}\right]^{1/2} \tag{6-4}$$

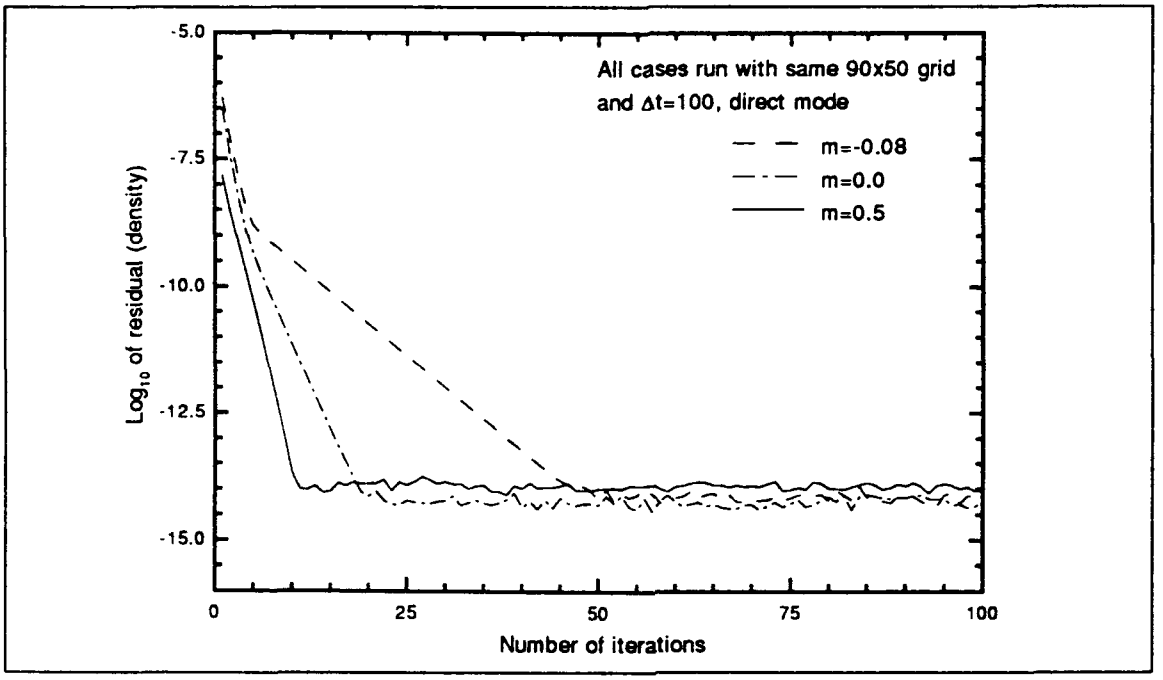

Figure 16. Dependence of convergence on the magnitude of the forcing function term

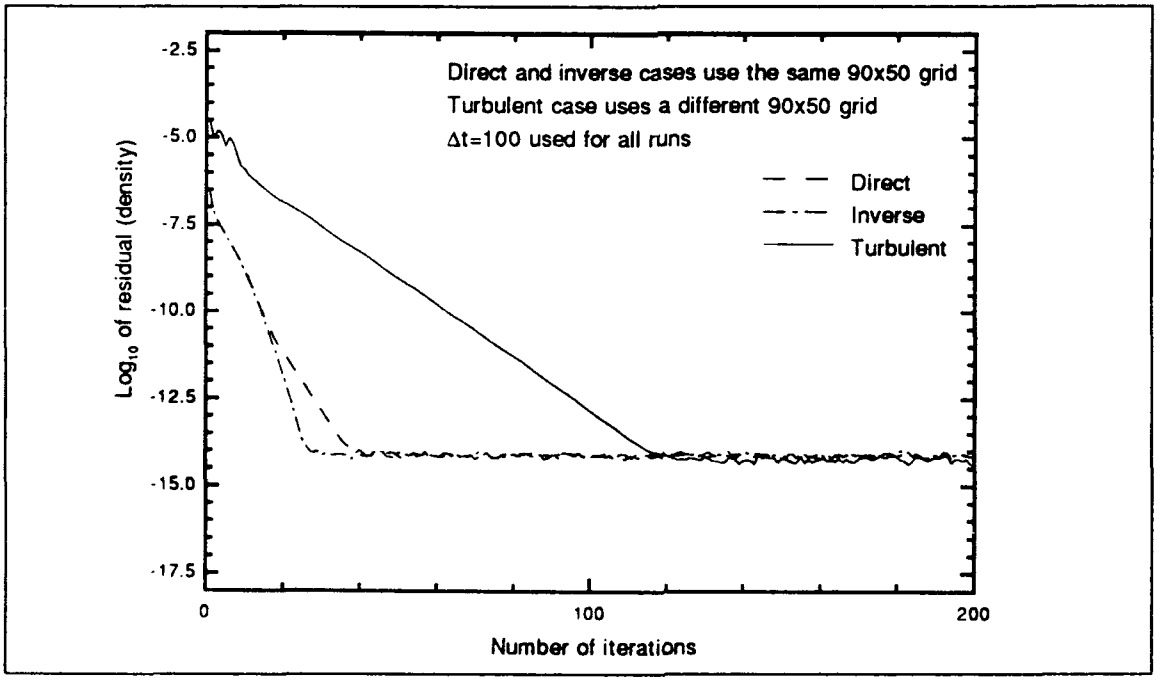

Figure 17. Effect of the momentum equation solver and viscous terms on convergence

$$n^+ = \frac{u_\tau n}{\nu} \tag{6-5}$$

Andersen's results are for $Re_\theta = 2570$, where Re_θ is the Reynolds number, based on the momentum thickness. In figure 15, the GBL results are compared for two profiles. For the first profile the Reynolds number is matched while for the second profile, the value of u^+ is matched at the upper edge of the flow. Fairly good agreement is obtained between the experimental data and the GBL results.

A Word on Convergence

The convergence characteristics of the boundary layer equations are difficult to quantify due to their non-linear character, the number of factors involved and the coupling between some of these same factors. The author has nevertheless gained sufficient experience with GBL to identify the major convergence characteristics of the code. Two basic mechanisms are at work. The first one is a convection mechanism which acts while a particular equation is being solved to update a given variable, e.g. $\tilde{u}^{n+1}$. At any given station, solution of the variable is affected by the upstream values of $\tilde{u}$ since the finite difference to its derivative involves upstream values of $\tilde{u}$. Recall that while solving this equation, all other variables are kept constant; only the values of the $\tilde{u}$ field are modified. The second mechanism is more difficult to identify. It involves the interplay between the different equations, i.e. how the solution of the momentum equation affects the solution of the energy equation, and so on.

Both mechanisms are affected by the sign and/or magnitude of the convective, viscous and time-step terms. Furthermore, the convective and viscous terms are coupled while the effect of the time-step terms is separate. Examples are now presented and discussed to indicate the role of the above factors.

Convective and viscous terms

Convective and viscous terms depend strongly on the streamwise momentum flow field. The former are function of the magnitude of the streamwise velocity while viscous terms are function of the normal gradient of the same quantity. As a result, the momentum equation is the driving force behind the overall iterative scheme. The evolution of the momentum flow field is itself driven by the forcing function, whether in the direct or inverse mode. Figure 16 shows the effect of the forcing term on convergence. It is observed that, at least for Falkner-skan flows, accelerated flows converge faster than decelerated flows. Separated flows are even slower to converge as demonstrated by the solution of the Klineberg flow; 400 iterations reduces the residual by only five orders of magnitude (the same time-step is used for all computations). Based on this result, one

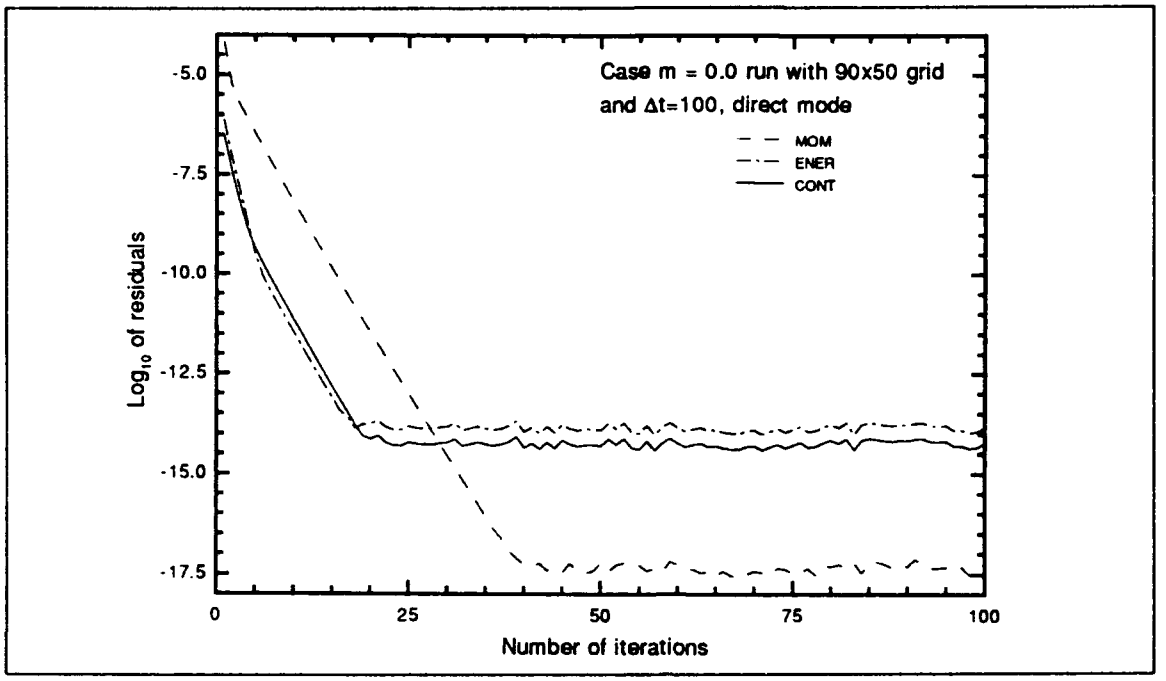

Figure 18. Manifestation of the convergence mechanism between the various equations

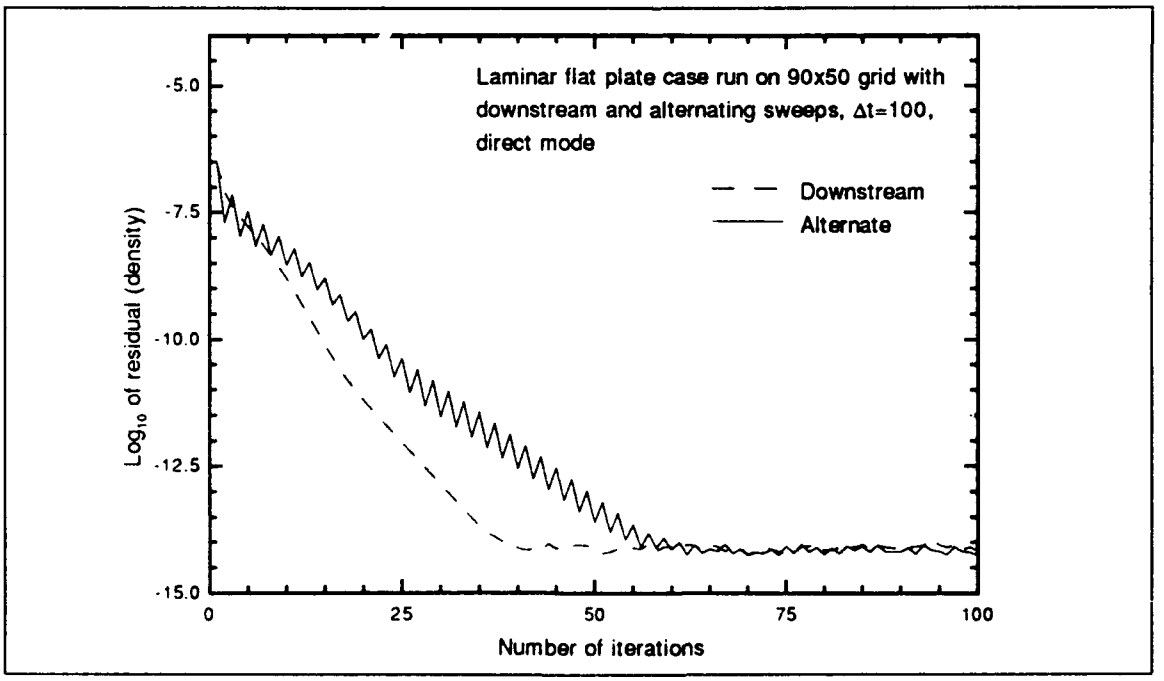

Figure 19. Effect of the direction of sweep on convergence - part 1

could speculate that, for airfoil flows, the regions of accelerated flow converge faster than regions of decelerated or separated flow. Hence, the overall convergence is governed by the regions of decelerated flow.

Recalling that different forcing functions are used for attached and reversed flow regions, hence how do we know if the solver affects the convergence? Figure 17 answers this question. The curves labelled *direct* and *inverse* show the convergence histories for the same flat plate flow solved using the direct and inverse solvers respectively. The two curves have similar convergence rates which supports the hypothesis that it is the magnitude and sign of the forcing function which drives convergence, not the method used to solve the momentum equation. The third curve of Figure 17 shows the convergence for the computation of the Andersen flow. It uses a different grid than the other curves, but it has the same time-step. It is included to demonstrate the effect of the turbulent terms.

The slope of the *turbulent* curve is less than for the laminar curve, which indicates that the turbulent terms decrease the convergence rate of the algorithm. These terms contribute to the magnitude of the elements of the left hand (or *implicit*) side of the equation only. The relative magnitude of the forcing term, located on the right hand (or *explicit*) side of the equation, is therefore decreased and the overall convergence is slower.

Another factor may be responsible for the slower convergence of the turbulent flow case with respect to the laminar flow cases: the quality of the initial guess. Assuming that convergence is rate limited, which it appears to be for a given time step and forcing function, more iterations are required to decrease the residual to machine zero when the initial guess is further away from the solution. For the turbulent case of figure 17, the initial guess is a laminar solution, which is quite different from the converged solution.

Figure 18 shows the typical relation between the momentum, energy and density residuals. The three variables converge to different values, which reflects the sensivity of each variable on the others, i.e. energy and density are less sensitive to changes of the velocity than velocity to changes of density or energy. More importantly, each equation requires a different number of iterations to converge with the convergence of the momentum flow field lagging that of the energy and density fields. This is a clear indication of the presence of a mechanism oetween the various equations.

Alternating the direction of sweep produces a sawtooth convergence history as illustrated in figure 19. The scheme converges while sweeping in the downstream direction, but diverges in the upstream direction, at least for the time-step used. In fact, it is the experience of the author that the scheme is unconditionally stable when sweeping in the downstream direction, but only conditionally stable when sweeping in the upstream direction.

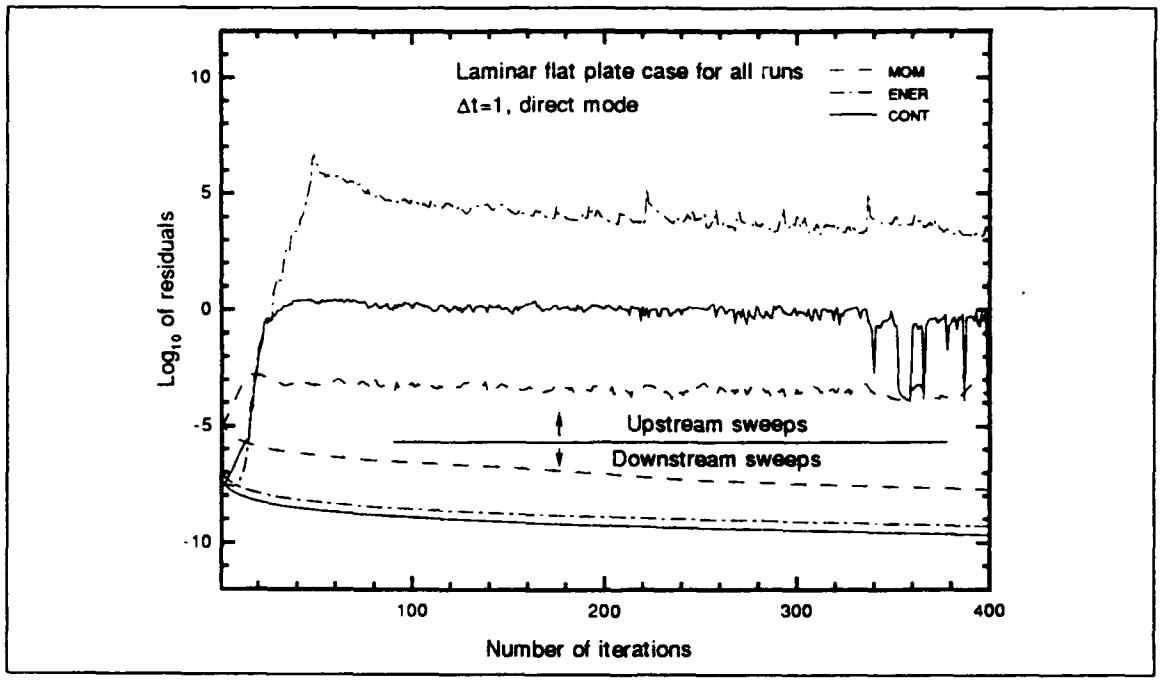

Figure 20. Effect of the direction of sweep on convergence - part 2

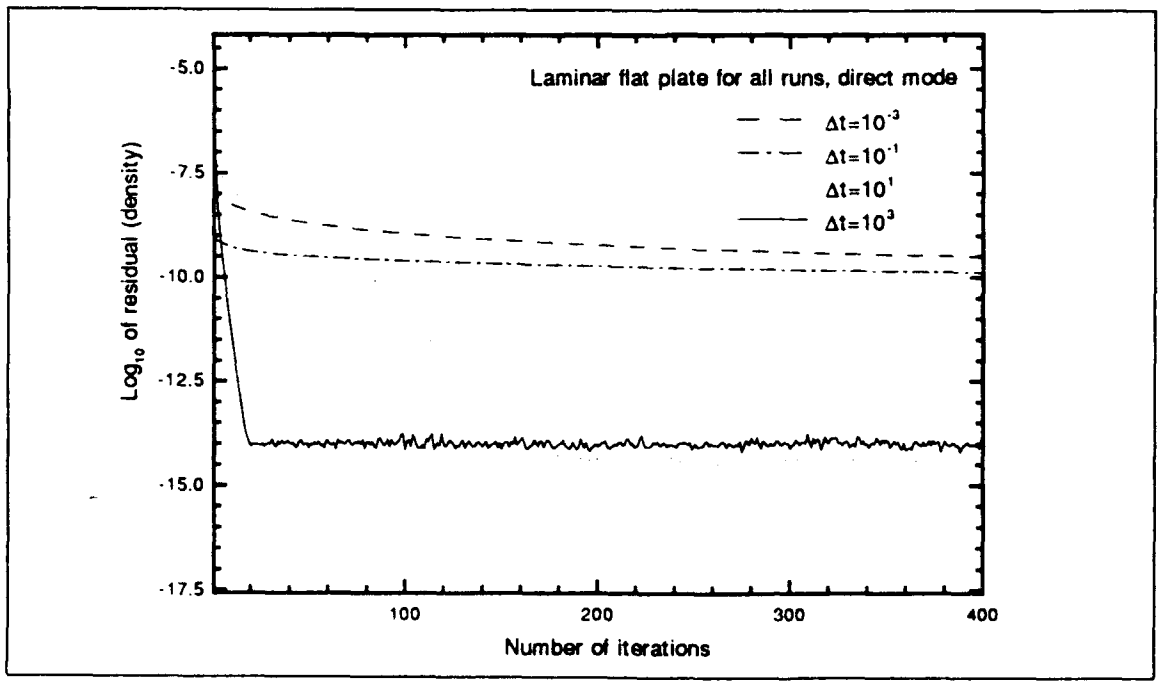

Figure 21. Effect of the time-step on convergence

For the flat plate case, the stability margin lies near the unit time-step value. Figure 20 compares the effect of downstream and upstream sweeps for this case. The downstream case converges slowly while the upstream case shows an increasing rate of divergence over the first fifty iterations, and then levels out to a large value. The flow field variables display an oscillatory behaviour. For smaller time-steps, the scheme converges, albeit very slowly (thousands of iterations) and at the cost of a loss in accuracy, as will be discussed in the "Time-step terms" subsection below.

Alternating the direction of sweep does not improve the convergence of the Klineberg case either. The reversed flow region is so small with respect to the attached flow field that any local convergence improvements cannot influence the overall convergence. Hence, there is no gain from alternating the direction of sweep. Then, why is the reversed flow region converging at all when all sweeps are in the downstream direction and the time-step is large? The answer resides in the magnitude of the contravariant velocity U. As a general rule, the magnitude of U is small in the separation bubble, thus the influence of the convective terms is reduced locally. One may speculate that it is this characteristic which allows the overall scheme to converge.

Time-step terms

The time term splits in two components, one on the implicit side of the equation, and one on the explicit side. On the implicit side, it increases the diagonal dominance of the system of equations. On the explicit side, it impresses the value of the flow field variable at the previous time step (i.e. $\tilde{u}^n$, $\tilde{H}^n$ or $(\tilde{\rho}\tilde{v})^n$) on the forcing function. For downstream sweeps of the laminar flat plate flow, the net effect of decreasing the time-step is to slow convergence, as shown in figure 21. As the magnitude of the time-step is decreased by orders of magnitude from 10^3 to 10^{-3}, more and more iterations are required to converge. Ultimately, a lower limit for the time-step is reached beyond which the accuracy is degraded because the time term overwhelms the forcing term and truncation error sets in.

Airfoil Case - Attached Flow

The ability of the code to compute attached flow conditions over an airfoil is examined by computing the flow over a NACA 0012 at $M_\infty = 0.7$ and $\alpha = 1.49°$. This test case, used by Holst[16] to compare several computer codes, produces attached flow over the full extent of the airfoil. It is mildly transonic with a maximum local Mach number of 1.04 near the 17% chord location.

The present results are for the standard configuration of the code with flow checks to detect regions of reversed flow, but without the algorithm to update the wall shear stress distribution in regions of reversed flow. The boundary layer grid is formed from a subset of the ARC2D grid and the initial flow data is from a converged ARC2D solution. However, the velocity data is multiplied by 0.98 to verify that GBL recovers the solution.

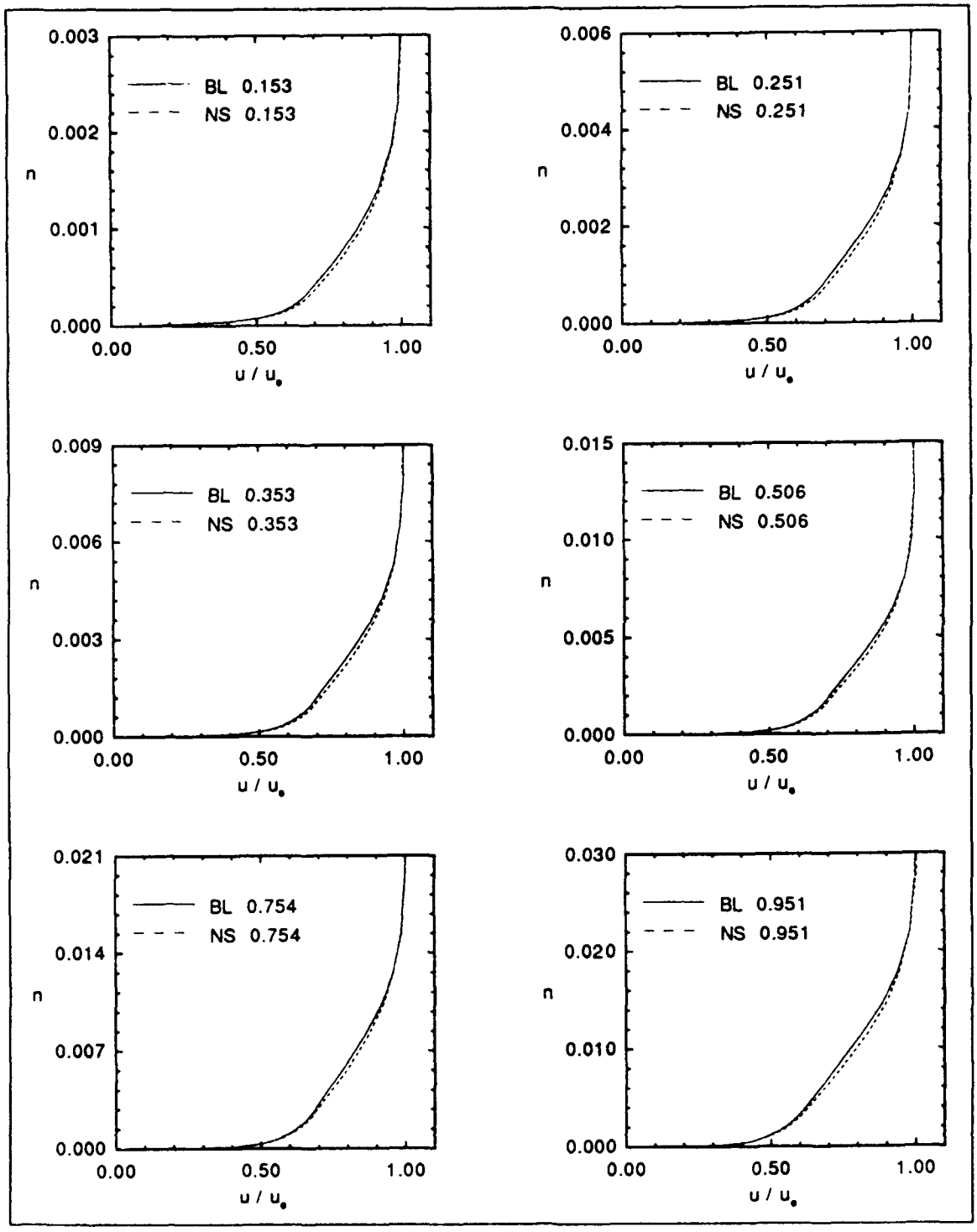

Figure 22. Comparison between GBL and ARC2D velocity profiles for attached flow case

Downstream sweeps are used to solve the equations and the solution is limited to the top surface of the airfoil in the interest of clarity. Note that the lower surface is easier to solve and does not generate new knowledge about the code. Finally, only airfoil stations are used since the $\eta = 1$ line does not overlap the wake centerline.

For the first set of results, the ARC2D surface pressure is used as the forcing function to GBL. The edge velocity, used as the upper boundary condition for the momentum equation, is computed from the velocity-pressure relation of equation (4-4). This case converges smoothly to machine zero in 125 iterations. Figure 22 compares the resulting velocity profiles to the ARC2D solutions at 15, 25, 35, 50, 75 and 95% of chord. Very good agreement is obtained for all stations except near the trailing edge where there is a small difference in edge velocity. This occurs because the code detects flow reversal over a very small range near the trailing edge, and switches to the inverse flow solver at these stations.

The reversed flow region is predicted because an appreciable normal pressure gradient exists near the trailing edge, but it is not accounted for by GBL. To correct for this condition, the pressure is interpolated from ARC2D at the approximate boundary layer edge. However, how do we find the edge? To answer this, we must consider the vorticity field around the airfoil. Recall from the airfoil initialization section that vorticity is present only within the boundary layer and that its intensity is largest near the surface. As we move out of the boundary layer in the normal direction, it decreases by several orders of magnitude until it vanishes. Figure 23 shows this variation for the present test case. It is observed that the curve with the value $|\omega| = 1$ corresponds closely to the boundary layer edge. The resulting pressure distribution is compared to the surface pressure in figure 24. The normal pressure variation near the trailing edge shows clearly. GBL does not predict separation with the new pressure distribution and the velocity profile results near the trailing edge are improved.

In another test run, the airfoil flow is solved using the inverse mode at all stations. The wall shear stress forcing function is obtained from a three point approximation of the Navier-Stokes data near the surface. This solution requires 35 more iterations than the direct mode case and the resulting velocity profiles are not accurate. Figure 25 shows that the edge velocity distribution is not captured. When run with the edge velocity matching option, the same case converges to the correct solution in 300 iterations. The larger number of iterations is required because each adjustment of the forcing function is equivalent to restarting the computations; an acceptable wall shear stress distribution is obtained in about 150 iterations. The matched edge velocity is within half of a percent and the resulting velocity profiles are essentially identical to those for the direct mode case. The final wall shear stress distribution, shown in figure 26, differs only slightly from the initial distribution, mostly in the supersonic bubble and immediately aft of the shock.

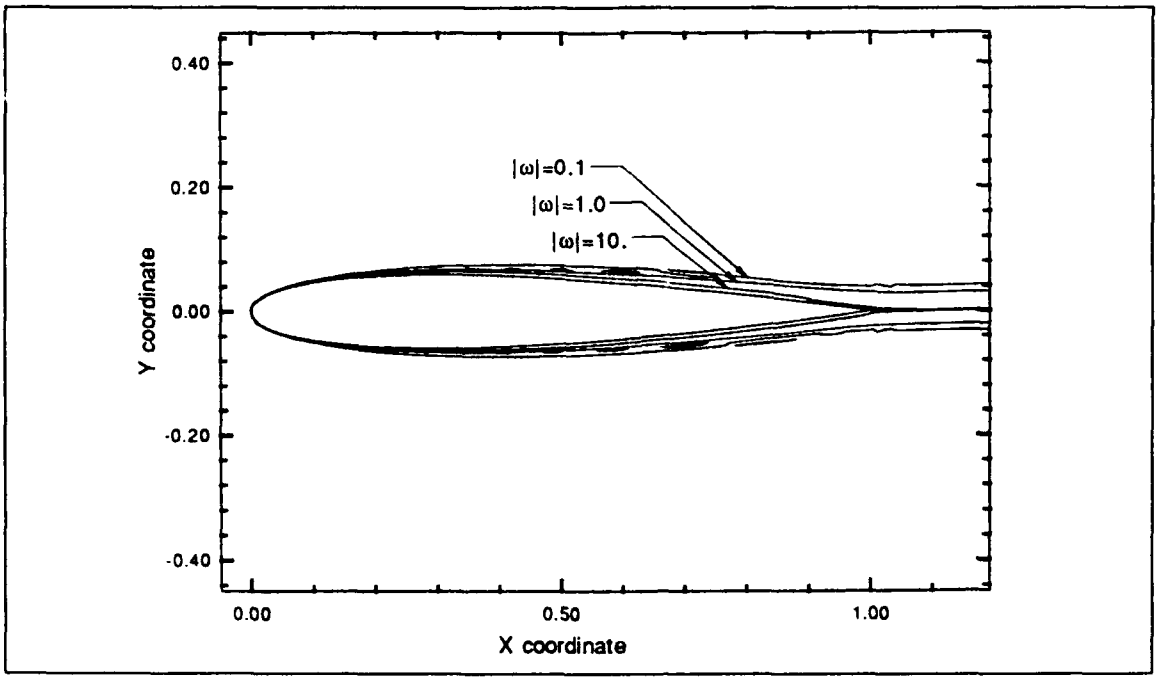

Figure 23. Vorticity contours of attached flow case extracted from ARC2D solution

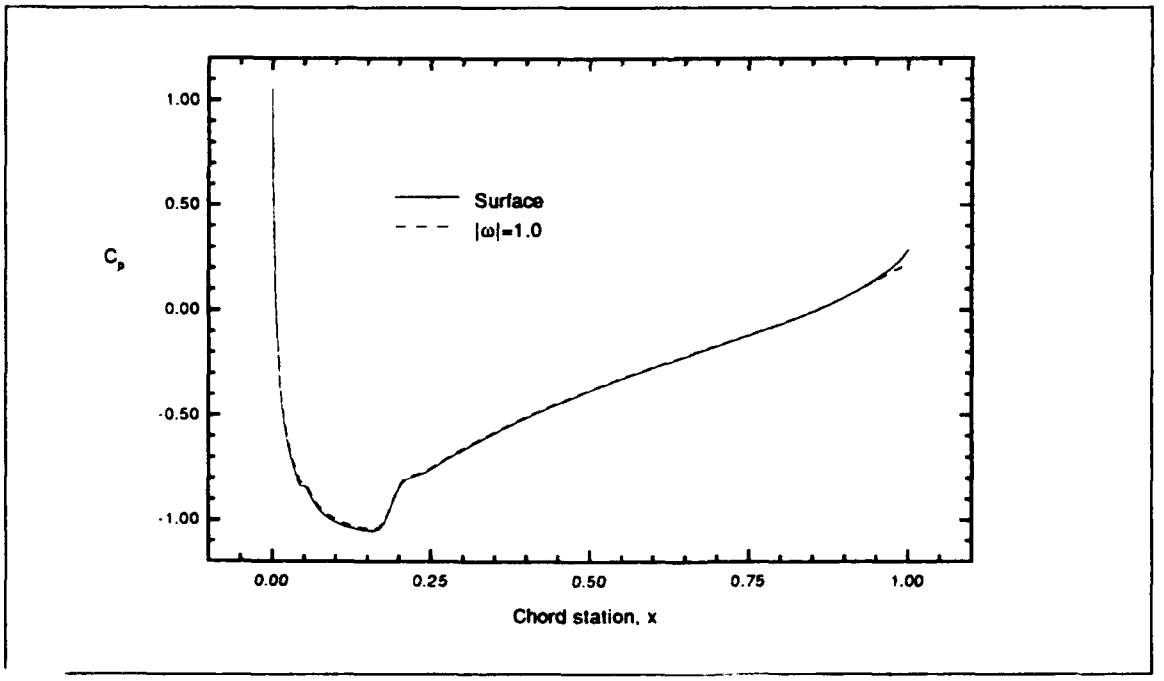

Figure 24. Comparison of surface pressure to pressure along vorticity line $|\omega| = 1$

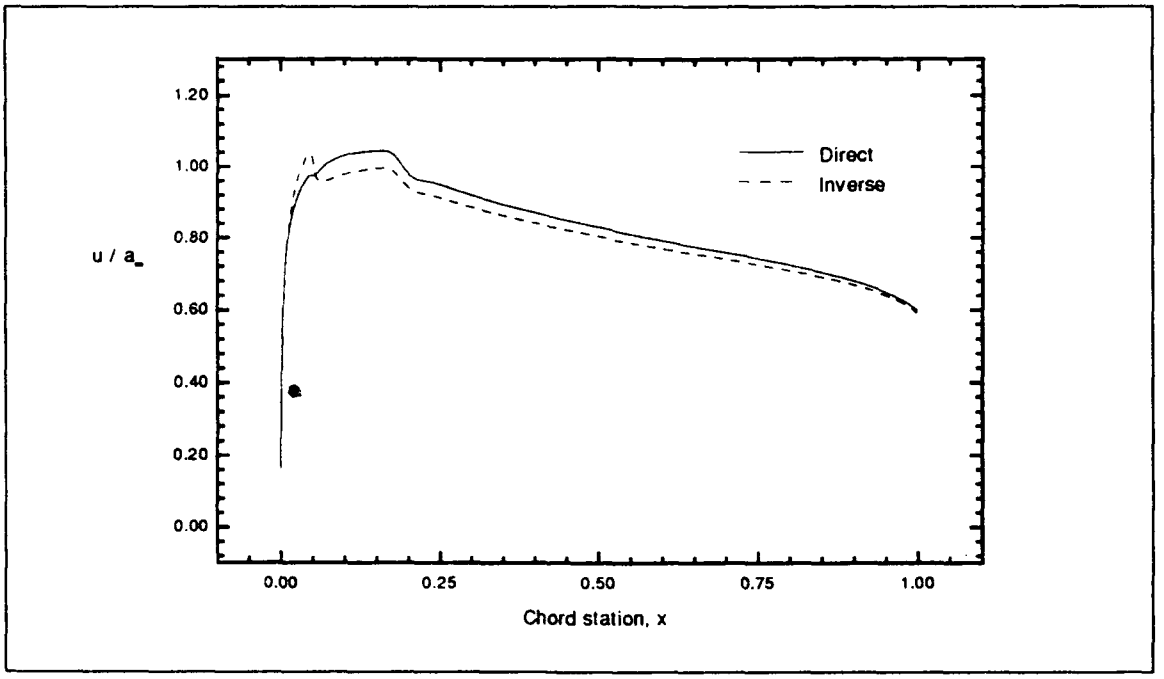

Figure 25. Direct and inverse mode values of edge velocity

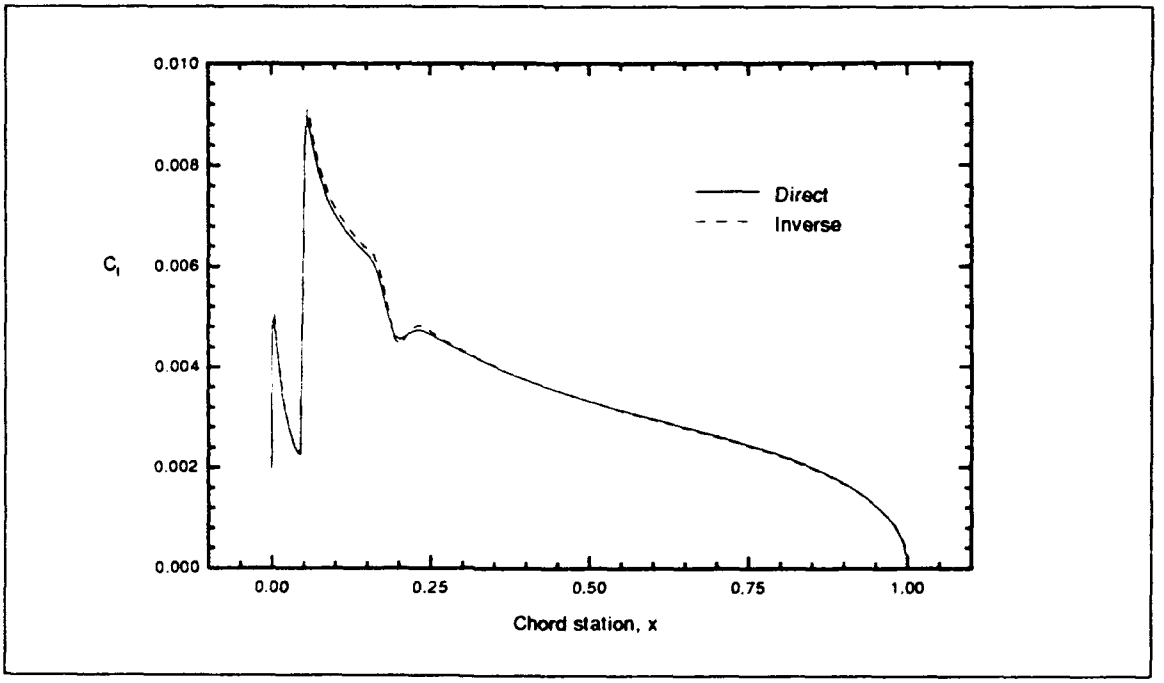

Figure 26. Initial and final wall shear stress distributions for match of edge velocity

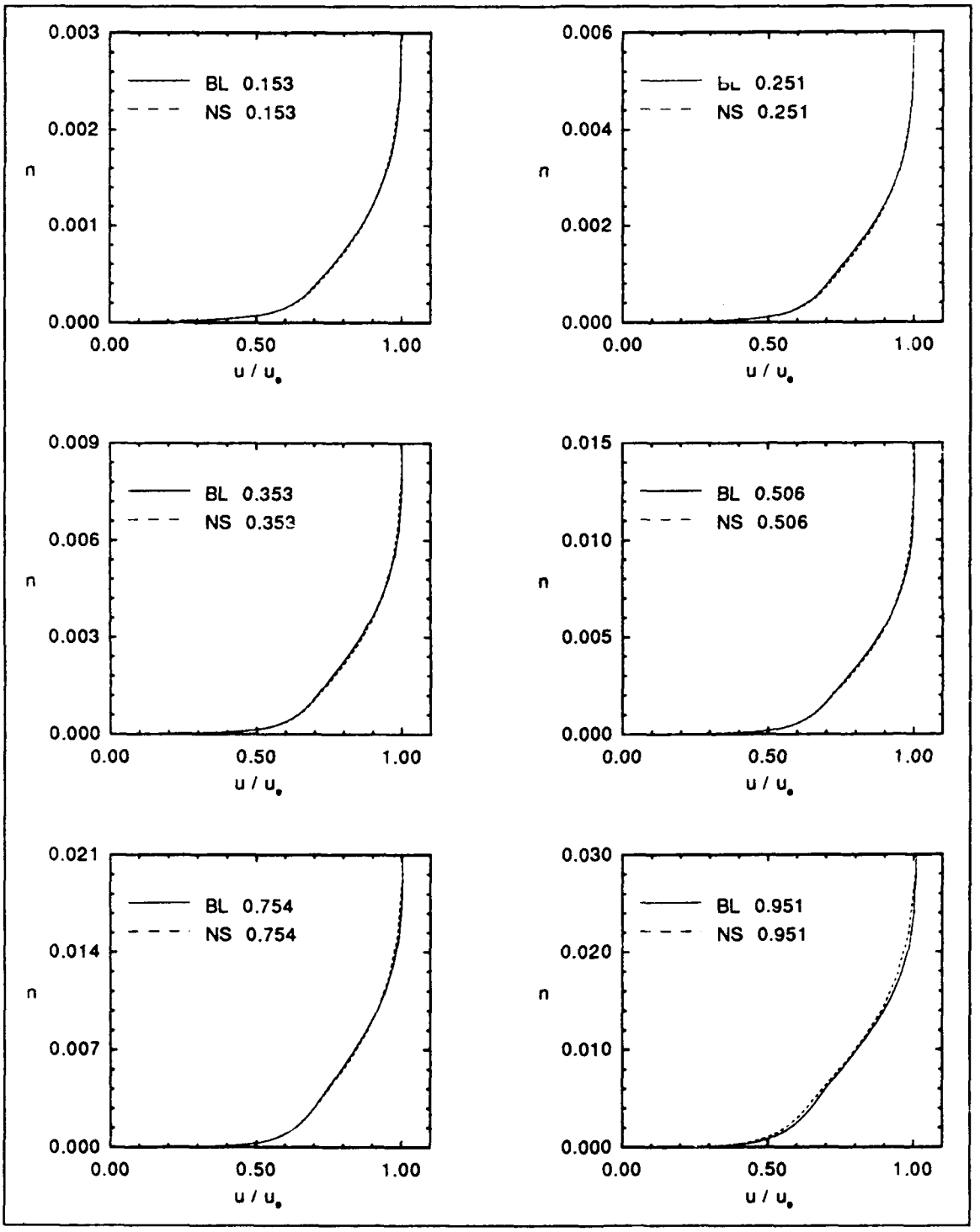

Figure 27. Velocity profiles for adaptive grid, direct mode

In the above inverse mode test case, one must be careful in selecting the pressure distribution to be matched. The wall shear stress forcing function is derived by applying the momentum equation at the surface and therefore, it represents surface pressure. Using the pressure distribution of the vorticity line $|\omega| = 1$, even though it is almost identical to the surface pressure (see figure 24), results in a substantial change ($\approx$25%) in the computed wall shear stress.

The above direct mode cases were run on a subset grid from ARC2D, but an "adaptive" grid can also be used. Such a grid is generated by GBL to overlap the ARC2D grid with the parameters $\Delta\tilde{n}_{\min} = 10^{-5}$, $f_l = 1.0$ and $f_u = 3.5$. The maximum number of points across the boundary layer is 45. Using this grid with the pressure distribution on the $|\omega| = 1$ line, GBL switches to the inverse mode for all stations aft of 60% chord immediately upon completing the first iteration. As a result, the edge velocity is not matched for these stations, but the velocity profiles are nevertheless acceptable.

The early switch to the inverse mode is traced to the corrupted data which yields poor estimates of the derivatives initially. It is corrected with a slow start procedure, i.e. under-relaxation factors are applied to the equations over the first ten iterations. For the first iteration, the under-relaxation factor is 10%, and it is increased by 10% upon each successive iteration. Figure 27 presents the resulting velocity profiles. They agree very well with those of ARC2D, although the normal pressure gradient effect can still be seen near the trailing edge. These results indicate that an improved adaptive grid scheme may benefit the code since all field points could be used; however, under-relaxation is required to start the computations to avoid tripping the inverse mode. The broad range of flow conditions may also make it difficult to find suitable criteria to adapt the grid.

Airfoil Case - Separated Flow

Two separated flow test cases are now presented. First, a NACA 0012 airfoil at 3° angle-of-attack and $M_\infty = 0.70$ is considered. This flow, previously computed by Maksymiuk et al.[17], accelerates rapidly over the leading edge to create a supersonic bubble with a maximum Mach number $M = 1.2$. It decelerates through a shock at $\approx$33% chord and immediately separates at the base of the shock. The separation, however, is small and the flow reattaches immediately up to a point near the trailing edge where it separates again. It is not clear that the trailing edge separation exists physically; it could be an artifact due to the local skewness of the grid. Finally, transition from laminar to turbulent flow is fixed at 5% chord.

A subset of Maksymiuk's Navier-Stokes grid is used as the boundary layer grid. The maximum number of points allowed through the boundary layer is 40. The computations are restricted to the airfoil upper surface with all sweeps done in the downstream direction. A Dirichlet condition is used at the upper η boundary of the momentum equation in direct mode regions while a Neumann condition is used for the inverse mode.

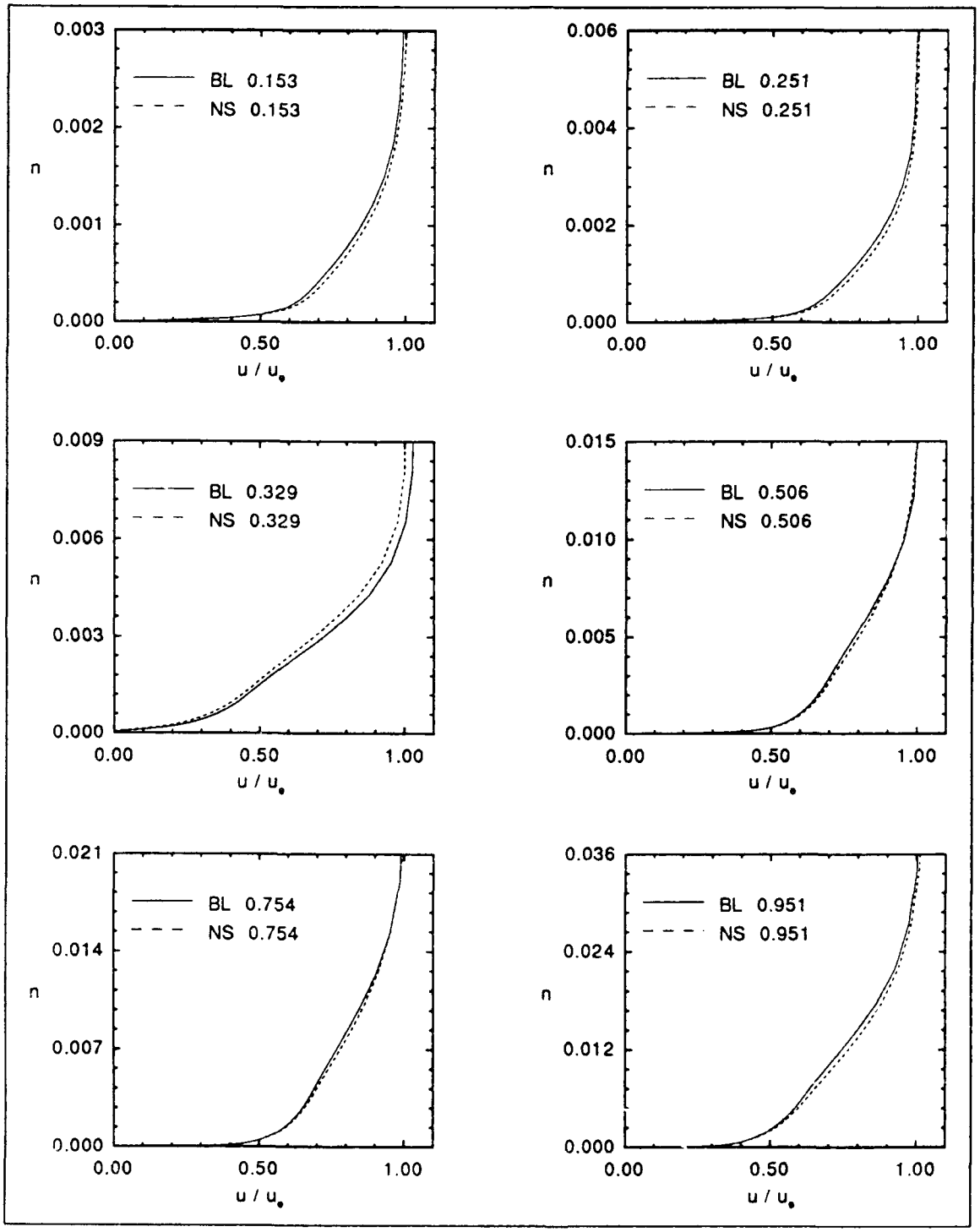

Figure 28. Velocity profiles for Maksymiuk case, velocity matching not enabled

Velocity matching is not enabled in the inverse regions and the forcing pressure distribution is from the vorticity level $|\tilde{\omega}_{cut}| = 1.0$.

The velocity profiles for the 15, 25, 33, 50, 75 and 95% chord stations are shown in figure 28 while figure 29 presents the corresponding edge velocity results. The edge velocity distribution is very close to the direct mode distribution, but it is still not fully captured around the shock and results in significant differences in the local velocity profiles. GBL still predicts good velocity profiles for most stations. Use of velocity matching in the inverse mode region results in a decrease of the $\tilde{\tau}_w$ forcing function aft of the shock (see figure 30) and a velocity defect of the local profiles as shown in figure 31. However, GBL recovers well and velocity profiles downstream of the shock compare well with those from the Navier-Stokes solution.

The matching process is slow and requires much user interaction. For this case, it is even doubtful that it fully works since it was required to relax the matching criteria UETOL to 1% to obtain convergence. In other words, the matched solution is not much better than the one without velocity matching. From these results, it appears that GBL has difficulty in handling shock waves. The code may have much to gain from the application of a Reimann problem at the shock, but this matter is not addressed in this work.

To further investigate the behaviour of GBL near shock waves, a second separated flow case is used. Proposed by Holst[16], it involves a NACA 0012 airfoil at 8.34° angle-of-attack and a freestream Mach number $M_\infty = 0.55$, which results in a very strong acceleration of the flow around the leading edge to a maximum Mach number $M = 1.34$, and a strong shock at 13% chord. The deceleration of the flow through the shock induces a substantial separation bubble at its base, followed by reattachment and a second separation zone at the trailing edge. The second separation bubble is due to the strong adverse pressure gradient present on the top surface of the airfoil. Laminar to turbulent transition is fixed at 5% chord for the computations.

To compute this flow, GBL is used without the velocity matching option and with the standard reverse zone detection algorithm. Dirichlet and Neumann boundary conditions are imposed at the upper η limit of the momentum equation for the direct and inverse modes, respectively, and the boundary layer grid is generated from a subset of the Maksymiuk Navier-Stokes grid with a maximum of 45 points in the normal direction. The vorticity level $|\tilde{\omega}_{cut}| = 0.5$ is used to determine the edge of the boundary layer. This value was selected after examination of the vorticity contours of the ARC2D solution, as depicted in figure 32. Note how the shock wave causes a large increase of the boundary layer thickness and a corresponding increase of the magnitude of vorticity.

The upper edge velocity is set from the vorticity level $|\tilde{\omega}_{cut}| = 0.5$, but surface pressure is used as the driving force. It is interesting to see, in figure 33, how much pressure varies across the boundary layer near the shock. The normal pressure gradient is already significant twenty points above the surface, which is still well within the boundary layer.

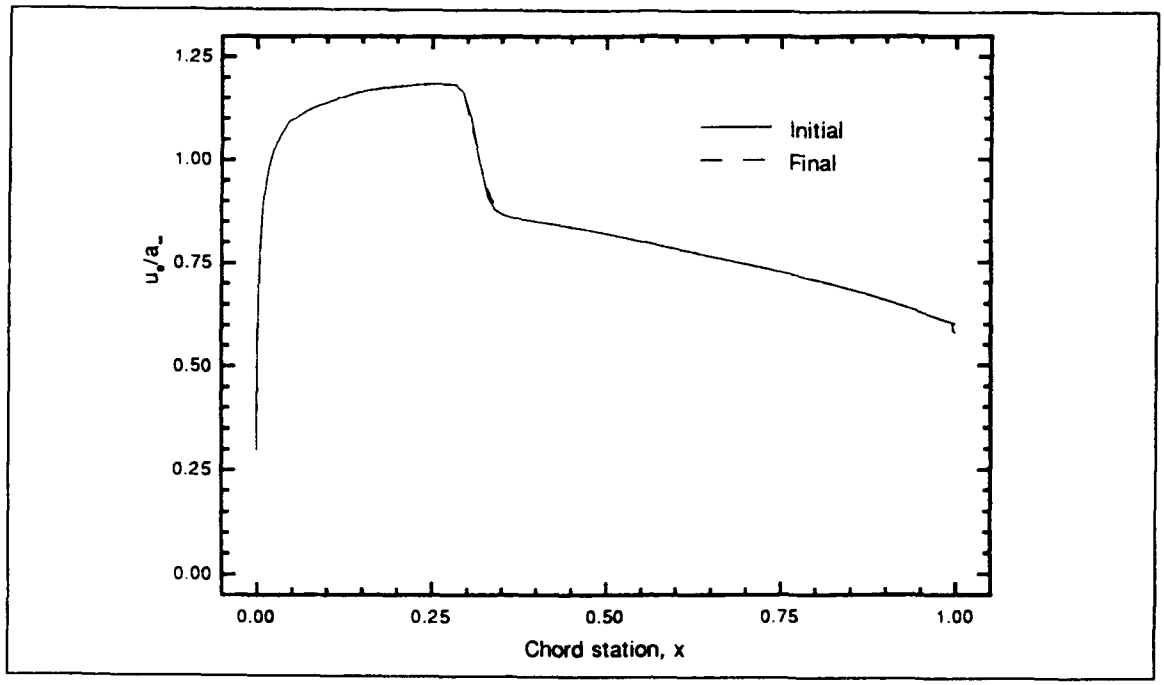

Figure 29. Comparison of edge velocity for Maksymiuk case, velocity matching not enabled

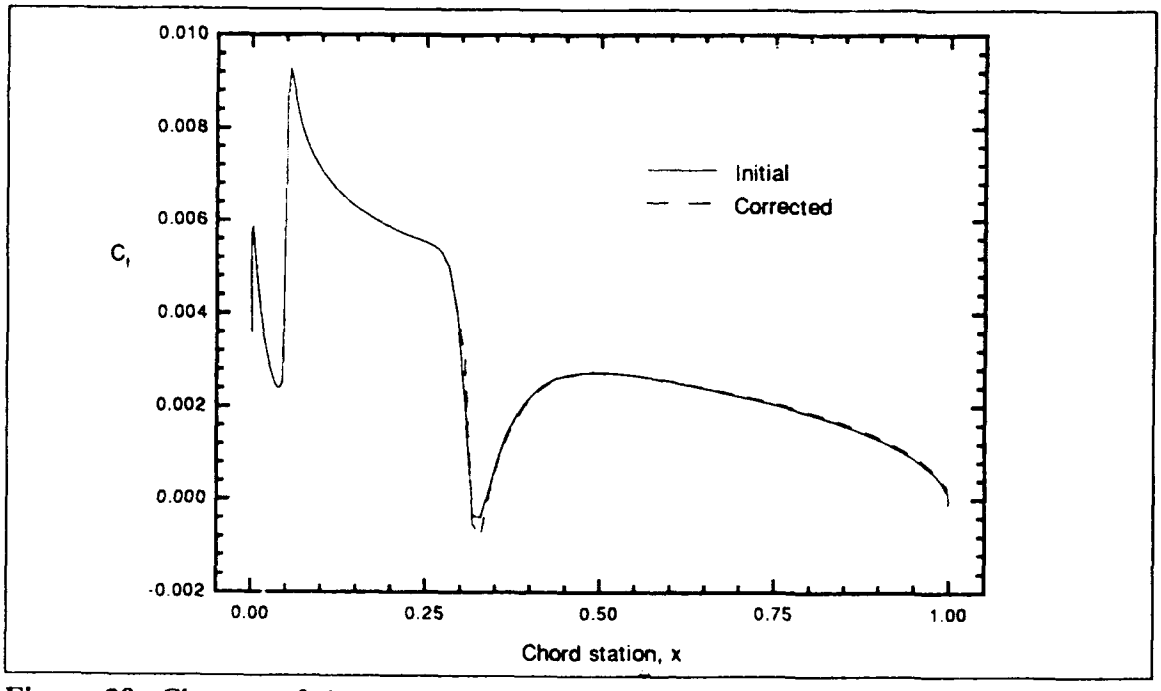

Figure 30. Change of inverse forcing function to match edge velocity with the Maksymiuk case

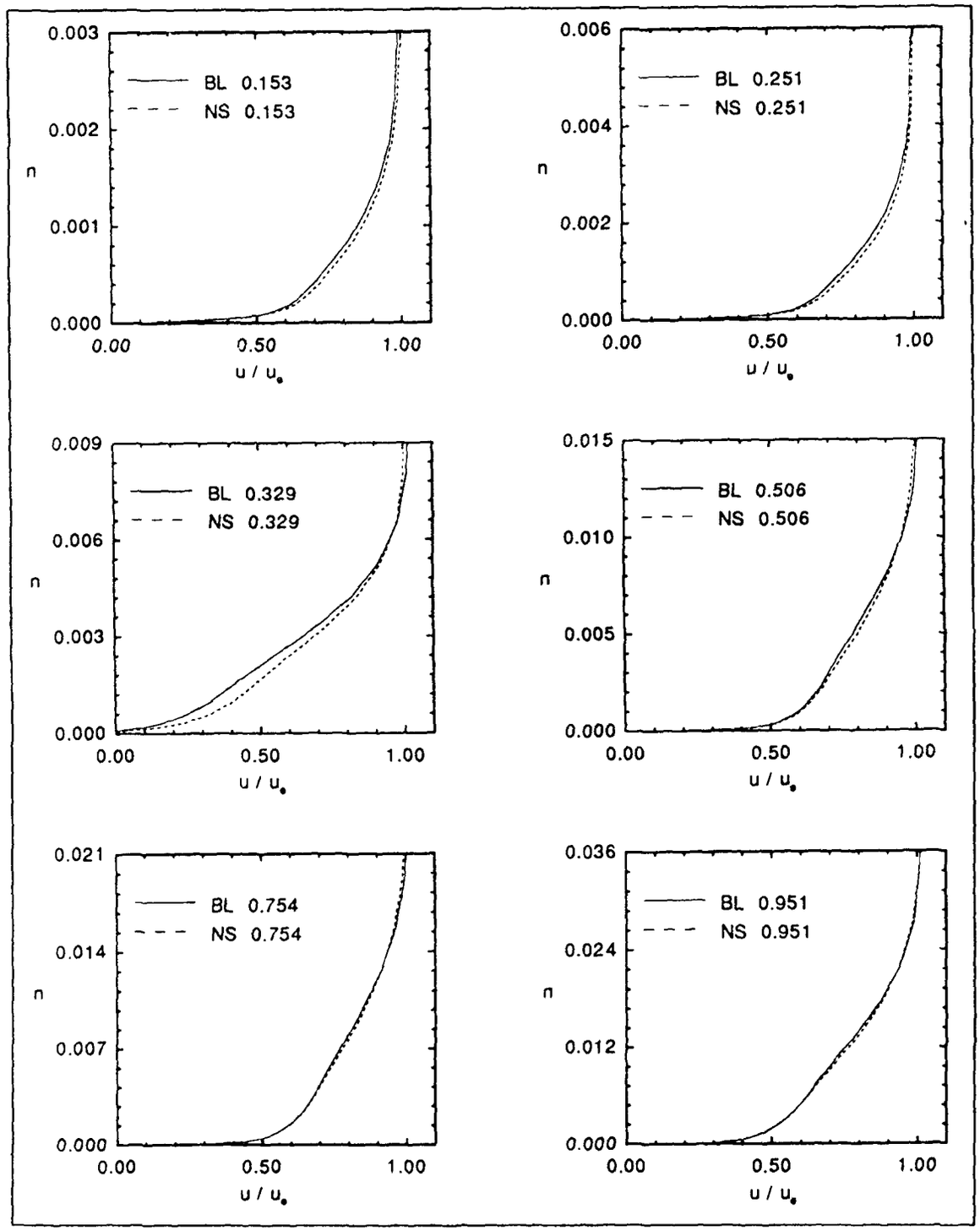

Figure 31. Velocity profiles for Maksymiuk case with edge velocity matching

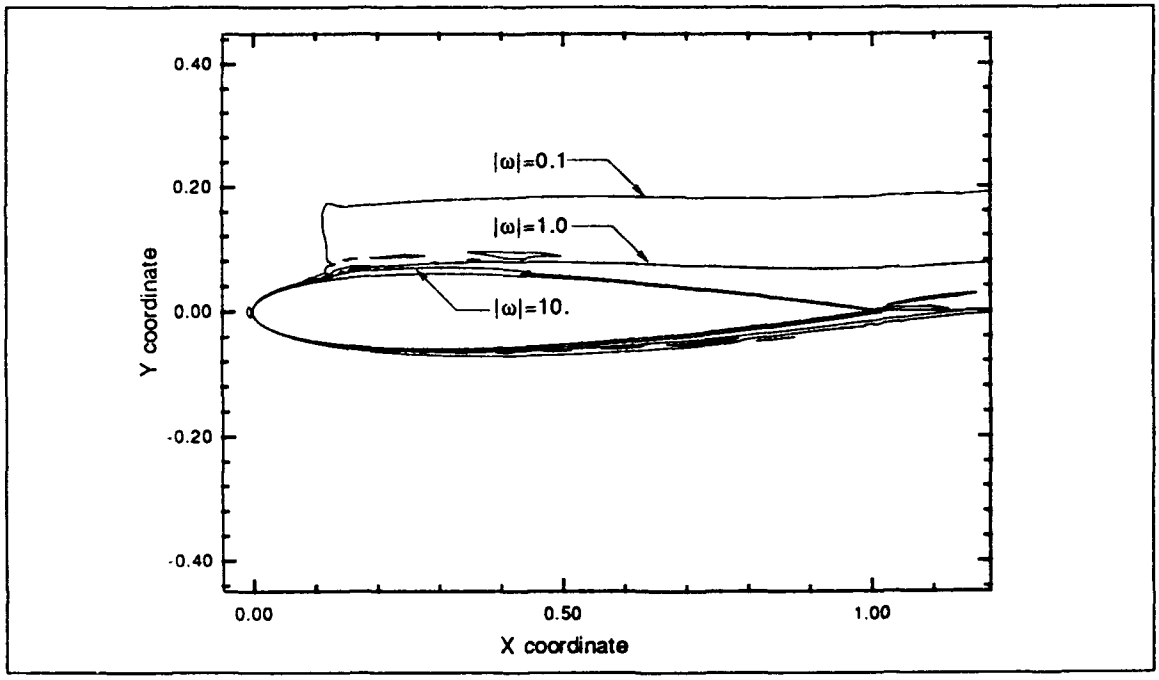

Figure 32. Vorticity contours for NACA 0012 airfoil at $\alpha = 8.34°$, $M_\infty = 0.55$

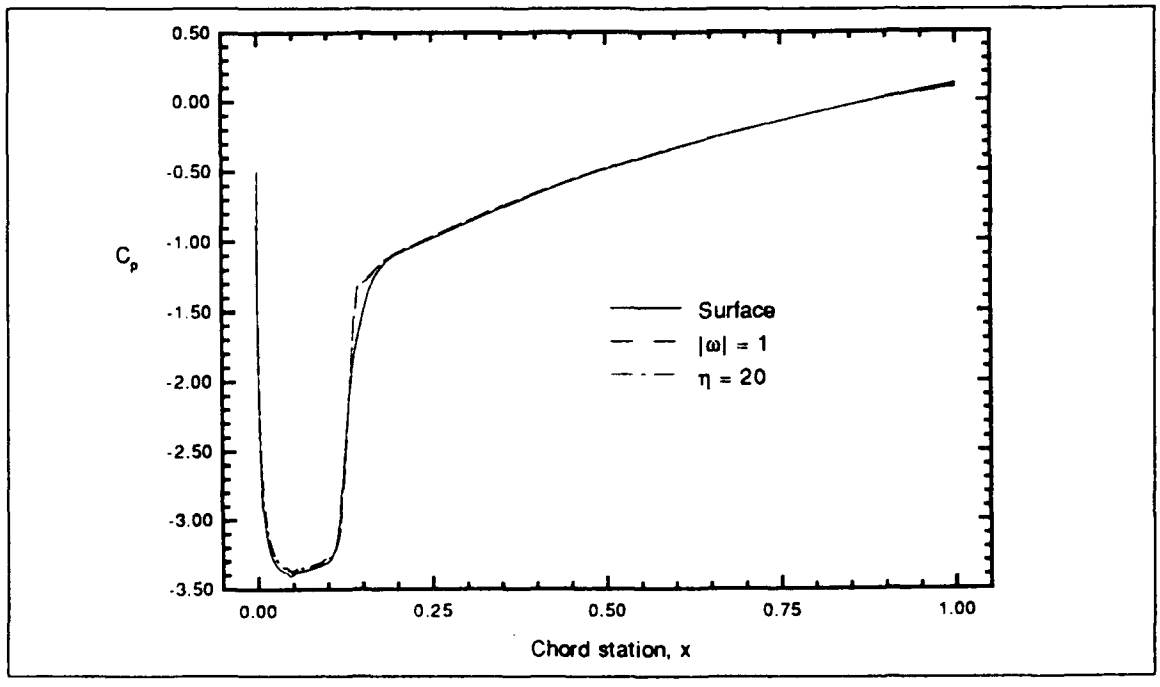

Figure 33. Variation of pressure across the boundary layer near the shock location

This pressure variation is responsible for the convergence problems of GBL near strong shocks and special handling of the shock region is required. This conclusion is also reached because GBL converges well and produces velocity profiles in good agreement with the Navier-Stokes solution when it is run with only stations ahead of the shock, or only station aft of the shock.

When the shock region is included, GBL produces the velocity profile results of figure 34. Note how a negative edge velocity is obtained for profiles near the shock wave. Recall that the surface pressure is used to drive the solution, hence this is a direct result of the normal pressure variation. These major changes to the profiles in the shock region propagate downstream. GBL recovers somewhat from this condition, but never completely; as it now thinks that the boundary layer edge is well within the grid limits. Conversely, it is possible that the grid is too large and the edge velocity condition is then taken well outside the range of validity of the boundary layer equations. This observation is supported by the good match of the GBL profiles with the Navier-Stokes solution near the wall for most stations aft of the shock location. The profiles in the trailing edge region disagree completely with the Navier-Stokes results, but this is not too surprising since the upstream velocity defect is compounded with a second separation zone in that region.

Interesting results are obtained when GBL is run with a grid made up of only the first twenty grid points in the normal direction. The edge velocity at the twentieth point is imposed as a Dirichlet boundary condition on the momentum equation for both the direct and inverse mode regions. The surface pressure is still used as the driving function. Figure 35 shows good agreement with the Navier-Stokes solution, even in the shock and trailing edge regions which are solved with the inverse mode. Use of GBL in this manner will be acceptable when it is integrated with the Navier-Stokes code to produce a Fortified Navier-Stokes procedure.

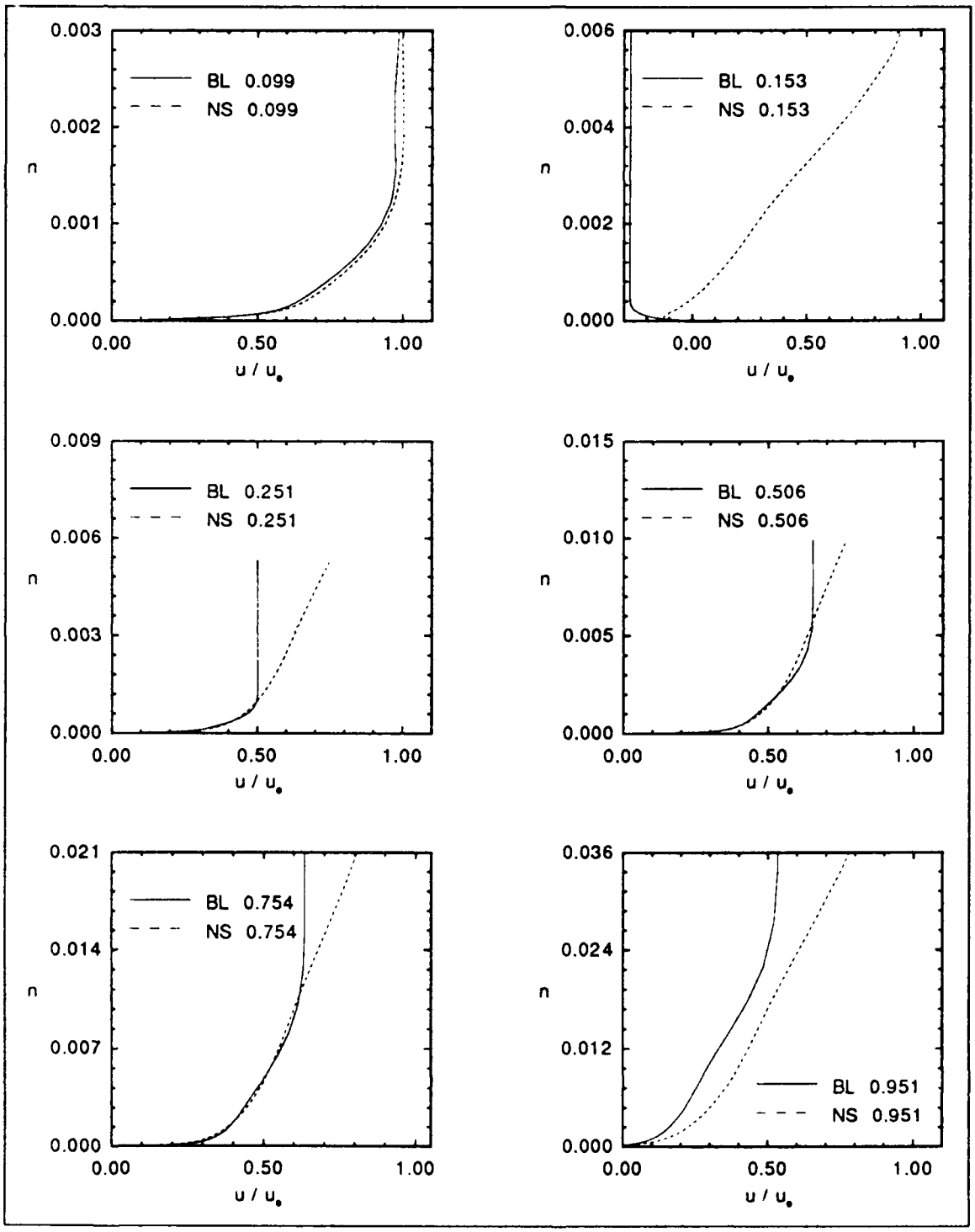

Figure 34. Velocity profiles for separated flow case of Holst - regular algorithm

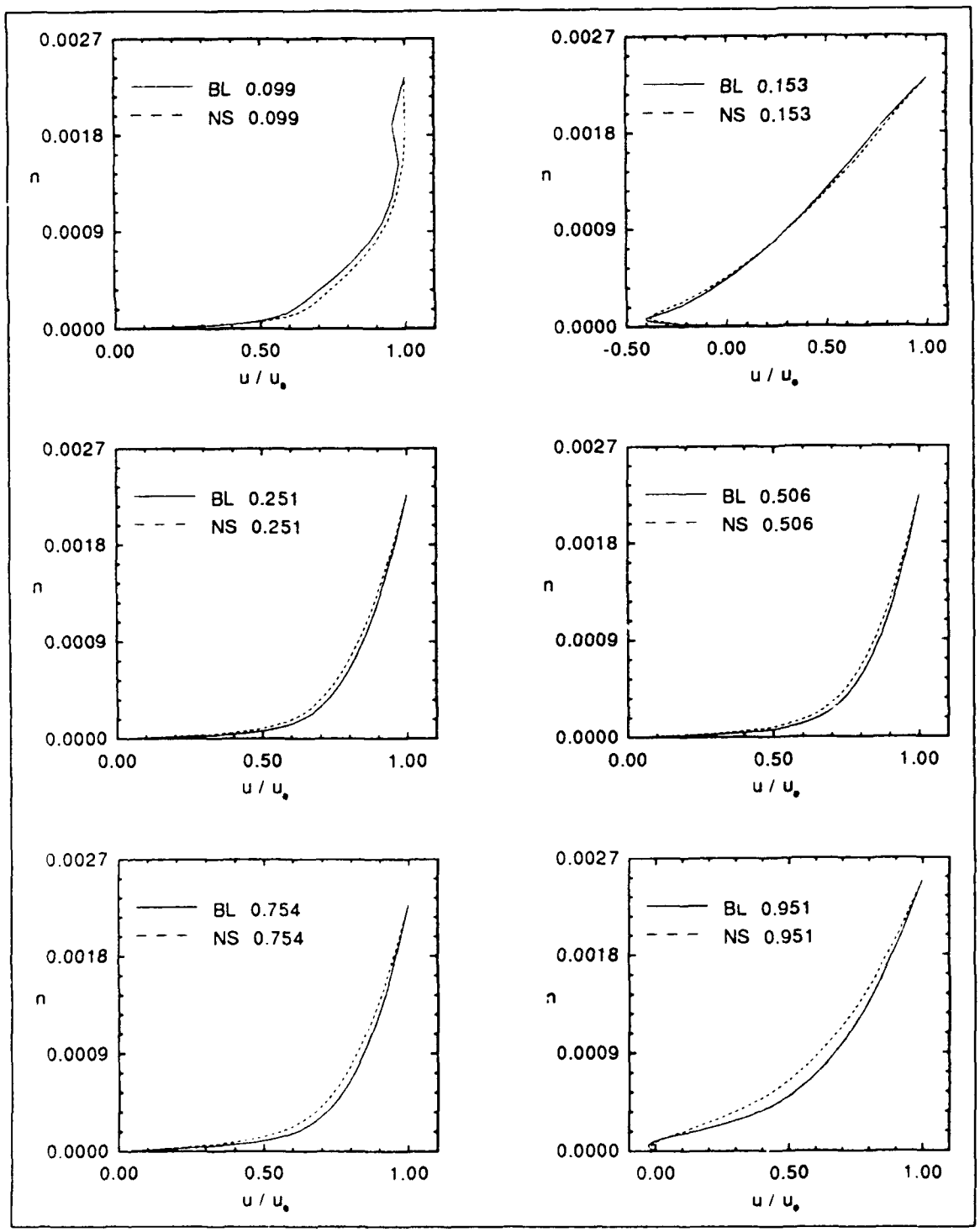

Figure 35. Velocity profiles for separated flow case of Holst - algorithm limited to twenty points

[BLANK PAGE]

CONCLUSIONS

The current work is aimed at achieving two principal goals. First, to develop a flexible research tool to study the behaviour of the Steger and Van Dalsem algorithm for solving the boundary layer equations, and second, to investigate how the same algorithm may be used in conjunction with a Navier-Stokes procedure to study the Fortified Navier-Stokes concept.

With respect to the first goal, GBL is now developed to the point of being a useful research tool. This report has outlined the main features of the code pertaining to its input requirements, its various initialization procedures, its solution techniques and miscellaneous computational tools required by the code. Verification results presented in the first part of the CODE TESTING section confirm that the algorithm is implemented correctly and that it is accurate for attached and separated laminar flow. It is also found that, at least for these simple flows, the algorithm converges faster when subjected to a negative pressure gradient and that it is slowest in regions of reversed flow. These results also show that the rate of convergence of the algorithm depends on the magnitude of the forcing function, and not on the method used to solve the equations, i.e. standard tridiagonal solver versus modified tridiagonal solver.

Simple numerical experiments with laminar flows show the algorithm to be unconditionally stable with respect to the time step used to relax the equations, provided the flow field is swept in the general downstream direction and that regions of reversed flow are weak. The algorithm is only conditionally stable when the flow field is swept in the opposite direction. Small time steps produce convergence, but the number of iterations required to obtain a converged solution is greatly increased in comparison to a downstream sweeping scheme. It should also be noted that too small a time step reduces the influence of the forcing function and degrades accuracy.

Comparison of GBL results against the experimental data of Andersen show that the Baldwin-Lomax turbulence model is succesfully implemented. It also demonstrates that selecting an initial guess far from the final answer affects convergence due to the maximum rate at which the method can converge for a given value of the forcing function.

Numerical results for the computation of transonic airfoil flows demonstrate the ability of GBL to solve such flows accurately using the direct, inverse or mixed direct/inverse modes. In particular, GBL predicts accurate results for fully attached flow conditions and for cases with embedded supersonic flow and a small separated flow region as long as the shock wave is weak. Strong shock waves cause an appreciable pressure gradient across the boundary layer which is not modelled in GBL; hence the present version of the code fails to compute such a case. The normal pressure gradient which characterizes the trailing edge region may also cause problems with GBL, but it is easily handled by selecting pressure from a point above the surface.

Significant progress has been made towards achieving the second goal of the present work: to develop a Fortified Navier-Stokes code. The framework to transfer GBL results to the Navier-Stokes code, and vice versa, is now in place. It is also shown that GBL predicts the velocity profiles accurately when a Dirichlet condition is applied at the upper boundary of the momentum equation with mixed direct/inverse modes, and the computations are limited to the portion of the grid nearest the wall (say 20 points).

Future work will concentrate on the integration of the GBL and Navier-Stokes codes and the development of suitable interaction mechanisms between the two procedures. The success of the method also hinges on the evolution of the boundary conditions and forcing functions in the Navier-Stokes procedure.

REFERENCES

1. Steger, J.L. and Van Dalsem, W.R., "Developments in the Simulation of Separated Flows using Finite Difference Methods", Proceedings of the 3rd Symposium on Numerical and Physical Aspects of Aerodynamic Flows, California State University, Long Beach, California, 1985.

2. Van Dalsem, W.R. and Steger, J.L., "Using the Boundary Layer Equations in Three-Dimensional Viscous Flow Simulation", Proceedings of the 58th Meeting of the AGARD Fluid Dynamics Panel Symposium on Applications of Computational Fluid Dynamics in Aeronautics, Aix-en-Provence, France, 1986.

3. Steger, J.L. and Van Dalsem, W.R., "Navier-Stokes and Viscous-Inviscid Interaction", NASA CP 3020, Vol. I, Part 2, Symposium held at NASA Langley Research Center, Hampton, Virginia, April 19-21, 1988.

4. Cebeci, T. and Smith, A.M.O., "Analysis of Turbulent Boundary Layers", Academic Press, 1974.

5. Van Dalsem, W.R. and Steger, J.L., "Simulation of Separated Transonic Airfoil Flow by Finite-Difference Viscous-Inviscid Interaction", Ph.D. Thesis, Stanford Univ., 1984.

6. Van Dalsem, W.R. and Steger, J.L., "Efficient Simulation of Separated Three-Dimensional Viscous Flows Using the Boundary Layer Equations", AIAA J.Vol. 25, no. 3, pp. 395-400, 1987.

7. Steger, J.L. and al., "A Formulation for the Boundary Layer equations in General Coordinates", NASA TM 100079, 1988.

8. Bergeron, M.D., "Finite Difference Form of the Compressible Boundary Layer Equations in Generalized Curvilinear Coordinates", Suffield Memorandum No. 1348, 1990.

9. Pulliam, T.H., "Efficient Solution Methods for the Navier-Stokes Equations", Von Karman Institute for Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computations, Brussels, Belgium, 1986.

10. Schlichting, H., "Boundary Layer Theory", Sixth edition, McGraw-Hill, 1968.

11. Baldwin, B.S. and Lomax, H., "Thin layer approximation and algebraic model for turbulent separated flows", Paper AIAA 78-257, presented at the AIAA 16th aerospace sciences meeting, Huntsville, Alabama, January 16-18, 1978.

12. Andersen, D.A., Tannehill, J.C. and Pletcher, R.H., "Computational Fluid Mechanics and Heat Transfer", Hemisphere Publishing Corporation, 1984.

13. V.M. Falkner and S.W. Skan, "Some approximate solutions of the boundary layer equations", Phil. Mag. 12, pp.865-896, 1931; ARC RM 1314, 1930.

14. Klineberg, J.M. and Steger, J.L., "The numerical calculation of laminar bo .dary layer separation", NASA TN D-7732, July 1974.

15. Andersen, P.S., Kays, W.M. and Moffat, R.J., "Experimental results for the transpired turbulent boundary layer in an adverse pressure gradient", J. Fluid Mech., vol. 69, part 2, pp. 353-375, 1975.

16. Holst, T.L., "Viscous transonic airfoil workshop compendium of results", AIAA paper No. 87-1460, presented at the AIAA 19^{th} fluid dynamics and lasers conference, June 8-10, Honolulu, Hawaii, 1987.

17. Maksymiuk, C.M., Swanson, R.C. and Pulliam, T.H., "A comparison of two central difference schemes for solving the Navier-Stokes equations", NASA technical memorandum 102815, July 1990.

APPENDIX A: Sample Input File

GBL Inputs		
Value	Name	Description
9000000.	RE	Freestream Reynolds number, Re_∞, based on freestream velocity u_∞
1.000	REFL	Reference length (in meters) for Re_∞
0.70	FSMACH	Freestream Mach number, M_∞
255.	FST	Freestream temperature in Kelvin
1	RSTURB	Turbulence model, 0 for laminar flow and 1 for Baldwin-Lomax model
0.05	TRANSLO	TRANSLO and TRANSUP indicate the chord stations where laminar to
0.05	TRANSUP	turbulent transition if fixed. Also used for the flat plate grid generation
3	RSINCO	Switch for initial conditions: 0 = FSK, 1 = Klineberg, 2,3 = Navier-Stokes.
0.10	DTA	Variables DTA and DTW allow the selection of different time steps
0.10	DTW	for the airfoil and wake stations.
200	ITRN	Number of iterations for run (1 to 2000)
0	RSUEDG	Type of initialization for u_e and p_e respectively, 0 = from surface conditions,
0	RSPEDG	1 = from interpolation at $\|\tilde{\omega}\|$ level $\|\tilde{\omega}_{\infty}\|$, 2 = from η line KBL
2	RSRES	Residue type, 0 = none, 1 = equation, 2 = time difference
1	RSGEOM	Indicate kind of geometry used. 0 = flat plate, 1 = ARC2D airfoil grid
15	NXS	Number of streamwise grid stations for flat plate grid generator
45	KBL	Number of normal points for flat plate and adaptive grid generators
0.01	XSGRD	First coordinate for flat plate grid generator, s_s
1.000	XEGRD	Last coordinate for flat plate grid generator, s_e
0.051	DXSGRD	Grid spacing between first and second ξ stations
0.00001	DYMGRD	Minimum η spacing for adaptive airfoil grid in INIT3
0.5	FLGRD	Multiplication factor for lower limit in grid generators
3.5	FUGRD	Multiplication factor for upper limit in grid generators
1.0	WCUT	Cutoff value of $\|\tilde{\omega}\|$ for interpolation of boundary layer edge conditions
.false.	LSYMIN	The value DYMGRD is imposed on the grid generator lower limit
.true.	LSGEDG	Used in INIT2, select to use NDT with $\|\tilde{\omega}\|$ criterion (or fix grid at KBL)

GBL Inputs (Continued)		
Value	Name	Description
.true.	LSMODE	The direct mode is used
.false.	LSFIXM	Only the LSMODE mode is used
.false.	LSPMAT	GBL attempts edge pressure/velocity match
.false.	LSSLOW	Slow start is used over the first 10 iterations
.false.	LSTGFS	The stagnation stations are reset to Falkner-Skan profiles
.false.	LSPNOR	The pressure varies in the normal direction
.true.	LSCRPT	The initial velocity is corrupted by CFACT
0.98	CFACT	Multiplication factor to corrupt initial velocity data
.false.	LSGRAP	The graphic option is enabled
1	RSWEEP	Sweep direction: -1 = $\xi_{max} \rightarrow \xi_{min}$, 0 = alternating, 1 = $\xi_{min} \rightarrow \xi_{max}$
.true.	LSXMOM	The x-momentum equation is enabled
.true.	LSUcUP	The contravariant velocity field U_c is updated
.true.	LSENER	The energy equation (total enthalpy) is enabled.
.true.	LSURHO	The density is updated
.true.	LSCONT	The continuity equation is enabled
.true.	LSURMM	The molecular viscosity is updated
.true.	LSURMT	The turbulent viscosity is updated
0.00	VM	Self-similar parameter for Falkner-Skan test case
190	JMINXI	Minimum station defining the computational field, ξ_{min}
352	JMAXXI	Maximum station defining the computational field, ξ_{max}
2	JOFFST	Offset about each side of the stagnation point
1	JZOFF	Number of stations added on each side of the inverse flow zones
0	BCDXL	Boundary conditions at the lower and upper η boundaries of the x-momentum equation in the direct and inverse mode, as well as for the energy equation. 0 = Dirichlet condition, 1 = 0^{th} order Neumann condition (slope is zero)
0	BCDXU	
0	BCIXL	
1	BCIXU	
1	BCDEL	
1	BCDEU	

GBL Inputs (Concluded)		
Value	Name	Description
1.00	WX	Under- or over-relaxation factor for the x-momentum and energy equations respectively
1.00	WH	
0.0001	UECHK(4)	$\tilde{u}_{NDT}$ variation below which $\tilde{\tau}_w$ is updated - for each zone
0.01	UETOL(4)	Convergence tolerance between $\tilde{u}_{NDT}$ and $\tilde{u}_e$ - for each zone
10	MAXUPD(4)	Maximum number of iterations between updates - for each zone
6	JFV	Number of stations for output of the velocity profiles (top surface only)
0.150	FVA(0-10)	Chord stations to output

APPENDIX B: GBL Direct and Inverse Solvers

GBL requires two matrix solvers for the direct and inverse mode, respectively. The direct mode solver, based on the Thomas algorithm, solves a tridiagonal system of equations, i.e. for a matrix system

$$\begin{bmatrix} B_{IL} & C_{IL} & & & & \\ A_{IL+1} & B_{IL+1} & C_{IL+1} & & & \\ & A_{IL+2} & B_{IL+2} & C_{IL+2} & & \\ & & & & & \\ & & & & A_{IU} & B_{IU} \end{bmatrix} \begin{bmatrix} X_{IL} \\ X_{IL+1} \\ X_{IL+2} \\ \vdots \\ X_{IU} \end{bmatrix} = \begin{bmatrix} D_{IL} \\ D_{IL+1} \\ D_{IL+2} \\ \vdots \\ D_{IU} \end{bmatrix}$$

where the coefficients A, B, C and D are provided by the calling routine, and X is the vector of unknows which is solved for. Upon returning, the solution X is in the D array. The array of diagonal elements B is also altered. The listing of subroutine THOMAS is given below.

```
C=======================================================================
C=======================================================================
      SUBROUTINE THOMAS(IL,IU,AA,BB,CC,DD)
C=======================================================================
C=======================================================================
C...  This routine is straight out of the Anderson textbook.  It is the
C...  standard Thomas algorithm.  Solution is stored in DD.
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AA(1),BB(1),CC(1),DD(1)
C.......................................................................
C...  Establish upper triangular matrix.
      LP = IL+1
      DO 10 I = LP,IU
        R = AA(I)/BB(I-1)
        BB(I) = BB(I)-R*CC(I-1)
        DD(I) = DD(I)-R*DD(I-1)
10    CONTINUE
C.......................................................................
```

```
C...  Back substitution.
      DD(IU) = DD(IU)/BB(IU)
      DO 20 I = LP,IU
        J = IU-I+IL
        DD(J) = (DD(J)-CC(J)*DD(J+1))/BB(J)
20    CONTINUE
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      RETURN
      END
```

The inverse mode solver, developed by Van Dalsem and Steger, uses a lower upper decomposition scheme to solve a system of equations of the form

$$\begin{bmatrix} B_{IL} & C_{IL} & & & \\ F_{IL+1} & B_{IL+1} & C_{IL+1} & & \\ F_{IL+2} & A_{IL+2} & B_{IL+2} & C_{IL+2} & \\ \vdots & & \ddots & \ddots & \ddots \\ F_{IU} & & & A_{IU} & B_{IU} \end{bmatrix} \begin{bmatrix} X_{IL} \\ X_{IL+1} \\ X_{IL+2} \\ \vdots \\ X_{IU} \end{bmatrix} = \begin{bmatrix} D_{IL} \\ D_{IL+1} \\ D_{IL+2} \\ \vdots \\ D_{IU} \end{bmatrix}$$

where the added coefficients F is due to the inverse forcing function. Once more, the solution is returned in array D. The Van Dalsem and Steger algorithm is implemented in subroutine INVTOM which is listed below.

```
C=======================================================================
C=======================================================================
      SUBROUTINE INVTOM(IL,IU,A,B,C,D,F)
C=======================================================================
C=======================================================================
C...  This routine implements the solution algorithm of Van Dalsem for
C...  quasi-tridiagonal matrices.  It uses an upper-lower decomposition
C...  followed by backsubstitutions for the solution.
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (KMAX=100)
      DIMENSION
```

```
     >  A(KMAX),        B(KMAX),        C(KMAX),        D(KMAX),
     >  F(KMAX),        G(KMAX),        P(KMAX),        R(KMAX)
C.....................................................................
C...  Compute the elements of the lower and upper triangular matrices.
      P(IU) = B(IU)
      R(IU) = F(IU)
      DO I = IU-1,IL+1,-1
        G(I) = C(I)/P(I+1)
        P(I) = B(I)-G(I)*A(I+1)
        R(I) = F(I)-G(I)*R(I+1)
      ENDDO
      G(IL) = C(IL)/P(IL+1)
      P(IL) = B(IL)-G(IL)*R(IL+1)
C.....................................................................
C...  Interim solution for multiplication by upper triangular matrix.
C...  Results are stored in array G.
      C(IU) = D(IU)
      DO I = IU-1,IL,-1
        C(I) = D(I)-G(I)*C(I+1)
      ENDDO
C.....................................................................
C...  Compute final solution.  The results are stored in array D.
      D(IL) = C(IL)/P(IL)
      D(IL+1) = (C(IL+1)-R(IL+1)*D(IL))/P(IL+1)
      DO I = IL+2,IU
        D(I) = (C(I)-R(I)*D(IL)-A(I)*D(I-1))/P(I)
      ENDDO
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      RETURN
      END
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

DEFENCE RESEARCH ESTABLISHMENT SUFFIELD
Box 4000, Medicine Hat, AB T1A 8K6

2. SECURITY CLASSIFICATION (overall security classification of the document. including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title.)

A RESEARCH CODE TO STUDY SOLUTIONS OF THE BOUNDARY LAYER EQUATIONS IN BODY CONFORMAL COORDINATES

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

BERGERON, Denis

5. DATE OF PUBLICATION (month and year of publication of document)

May 1991

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

85

6b. NO. OF REFS (total cited in document)

17

6. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Suffield Memorandum No. 1353

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

SM 1353

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

---

11 DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availabilty (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

NIL

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

DCD03 2/06/87

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both offical languages unless the text is bilingual).

This report describes the development of a computer code to solve the two-dimensional boundary layer equations in direct, inverse or mixed direct/inverse mode for airfoil flows. The solution algorithm uses body conformal coordinates and a relaxation scheme with flow-dependent operators. The code incorporates two methods to sweep the flow field.

Topics include a description of the code structure, its input requirements, boundary condition and flow field initialization and various software tools required by the main algorithm. Finally, verification results are presented.

This work, done in cooperation with the University of Toronto, seeks the development of a boundary layer code compatible with the NASA Ames ARC2D Navier-Stokes code. The two codes will be used in a study of the Fortified Navier-Stokes concept.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

COMPUTATIONAL FLUID DYNAMICS

BOUNDARY LAYERS

BODY CONFORMAL COORDINATES